514-3ABS/0197

MISSION OPERATIONS AND DATA SYSTEMS DIRECTORATE

Landsat 7 Processing System (LPS)
As-Built Specification (ABS)

November 1997

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland

Landsat 7 Processing System (LPS)
As-Built Specification (ABS)

November 1997

Prepared by:

Edward Criscuolo Jr 11/5/97

Ed Criscuolo, Software Task Leader — Date
Landsat 7 Processing System, CNMOS, Computer Sciences Corp.

Robert Schweiss 11/6/97

Robert Schweiss, Systems Engineering Manager — Date
Landsat 7 Processing System, Code 514

Quality Assured by:

Sheila Whisonant 11/5/97

Sheila Whisonant — Date
Landsat 7 Processing System, CNMOS, Computer Sciences Corp.

Concurred by:

Nathaniel E Daniel 11-5-97

Nate Daniel, Element Manager — Date
Landsat 7 Processing System, CNMOS, Computer Sciences Corp.

Approved by:

Joy Henegar 11-6-97

Joy Henegar, Project Manager — Date
Landsat 7 Processing System, Code 514

Goddard Space Flight Center
Greenbelt, Maryland

Abstract

The Landsat 7 Processing System (LPS) provides Landsat 7 data receipt and processing support to the Landsat 7 program. The LPS receives raw wideband data from the Landsat Ground Station located at the EROS Data Center (EDC); processes it into Level 0R, browse, and metadata files; and provides the files to the EDC Distributed Active Archive Center, also located at the EDC. The detailed design presented in this document is based on the information contained in the LPS functional and performance specification, LPS system design specification, LPS operations concept document, and LPS software requirements specification.

Keywords: *functional and performance specification (F&PS), Landsat 7, Landsat Ground Station (LGS), EROS Data Center (EDC) Distributed Active Archive Center (DAAC), Landsat 7 Processing System (LPS), Mission Operations and Data Systems Directorate (MO&DSD), Mission Operations and Systems Development Division (MOSDD), Systems Management Policy (SMP)*

Preface

This document is under the control of the LPS Project Configuration Management Board (PCMB).

Configuration change requests (CCRs) to this document, as well as supportive material justifying the proposed change, shall be submitted to the LPS PCMB. Changes to this document shall be made by document change notice (DCN) or by complete revision.

Address questions and proposed changes concerning this document to:

Robert Schweiss, Systems Engineering Manager
Landsat 7 Processing System
Code 514
Goddard Space Flight Center
Greenbelt, MD 20771

Contents

Section 1. Introduction

Section 2. Detailed Design Overview

Section 3. Operational Scenarios

Section 4. Global Libraries

Section 5. Raw Data Capture Subsystem

Section 6. Raw Data Processing Subsystem

Section 7. Major Frame Processing Subsystem

Section 8. Payload Correction Data Processing Subsystem

Section 9. Image Data Processing Subsystem

Section 10. Management and Control Subsystem

Section 11. LPS Data Transfer Subsystem

Section 12. Database Design

Section 13. User Interface

Tables

Figures

Section 1. Introduction

1.1 Purpose

This as-built specification (ABS) describes the detailed design of the Landsat 7 Processing System (LPS) and supports the implementation of the LPS. Throughout this document, the term "LPS" may be referred to as "the system."

1.2 Scope

The LPS is organized into four primary software configuration items (SWCIs): a global library, subsystem libraries, a relational database, and a user interface. This document covers the detailed design of each one.

The detailed design is derived from the LPS software requirements specification (SRS), LPS system design specification (SDS), LPS functional and performance specification (F&PS), and the LPS operations concept document. The LPS also incorporates various technical studies completed by the LPS project.

1.3 Applicable Documents

The following documents contain additional details regarding the Landsat 7 project, the LPS, and the external systems interfaced by LPS.

1.3.1 Specification Documents

The following documents provide the basis for developing the LPS detailed design presented in this ABS:

1. National Aeronautics and Space Administration (NASA) Goddard Space Flight Center (GSFC), Mission Operations and Data Systems Directorate (MO&DSD), *Landsat 7 Processing System (LPS) Software Requirements Specification*, Revision 1, 560-8SWR/0195, April 28, 1995

2. —, *Landsat 7 Processing System (LPS) Functional and Performance Specification*, Revision 1, 560-8FPS/0194, July 31, 1996

3. Lockheed Martin Astro Space (LMAS), *Landsat 7 System Data Format Control Book (DFCB), Volume 4 , Revision D—Wideband Data*, 23007702, February 11, 1997

4. NASA/GSFC, *Interface Control Document (ICD) Between the Landsat 7 Ground Station (LGS) and the Landsat 7 Processing System (LPS)*, Signoff Copy, 560-11CD/07694, October 17, 1996

5. —, *Interface Control Document Between the EOSDIS Core System (ECS) and the Landsat 7 System*, Working Draft, 194-219-SE1-003, August 1, 1994 (Note: Includes LPS-EDC DAAC interface requirements.)

6. —, *Memos of Understanding Between the Landsat 7 Processing System and the Mission Operations Center (MOC)*, Draft, May 1995

7. —, *Interface Control Document between the Landsat 7 Processing System and the Image Analysis System (IAS)*, Revision 1, 515-11CD/0195, July 29, 1996

8. Consultative Committee for Space Data Systems (CCSDS), *Recommendation for Space Data System Standards; Advanced Orbiting Systems (AOS), Networks and Data Links: Architectural Specification, Blue Book*, CCSDS 701.0-B-1, Issue 1, October 1989

9. Computer Sciences Corporation (CSC), *SEAS System Development Methodology*, July 1989

10. NASA/GSFC, MO&DSD, *Landsat 7 Processing System (LPS) Operations Concept*, Signoff Copy, 560-3OCD/0194, April 15, 1996

11. LPS/MO&DSD, *Landsat 7 Processing System (LPS) Interface Definitions Document*, Review, 514-3IDD/0195, October 20, 1995

12. —, *Landsat 7 Processing System (LPS) Output Files Data Format Control Book*, 510-3FCD/0195, November 14, 1996

1.3.2 Reference Documents

The following documents are used as additional sources of background information for the implementation of the LPS software requirements:

1. NASA/GSFC, MO&DSD, *Systems Management Policy*, MDOD-8YMP/0485, July 1986

2. NASA/GSFC, *Landsat 7 Detailed Mission Requirements*, May 15, 1995

3. NASA/GSFC, MO&DSD, *Landsat 7 Ground Station (LGS) Functional and Performance Specification (F&PS)*, Review, 531-FPS-GN/Landsat 7, January, 10, 1996

4. MO&DSD, *Mission Operations Concept Document for the Landsat 7 Ground System*, December 5, 1996

5. Martin Marietta Astro Space (MMAS), *Landsat 7 Image Assessment System (IAS) Operations Concept*, Landsat 7 Library No. 5527, December 19, 1994

6. NASA, *Landsat 7 Level 1 Requirements*, Draft, August 8, 1994

7. United States Geological Survey (USGS) National Oceanic and Atmospheric Administration (NOAA), *Index to Landsat Worldwide Reference System (WRS) Landsats 1, 2, 3, and 4*, 1982

8. MMAS, *Landsat 7 System Program Coordinates System Standard*, proposed update draft, 23007610A, August 1994

9. NASA/GSFC, *LPS Software Management Plan*, June 1, 1995

10. —, *LPS Project Management Plan*, May 11, 1995

11. —, *Landsat 7 Detailed Mission Requirements*, January 1, 1995

12. —, *Landsat 7 System and Operations Concept* (pre-CCB baseline version), 430-11-06-003-0, July 1, 1994

13. Mission Operations and Data Systems Engineering, *Renaissance Catalog of Building Blocks*, Preliminary, January 26, 1995

14. CSC/SEAS, *WRS Scene Identification Algorithms*, Draft, February 1, 1995

15. —, *Landsat 7 Processing System (LPS) Users Guide*, 514-3SUG/0195R2.0, July 1997

Section 2. Detailed Design Overview

This section provides an overview of the LPS detailed design. It discusses the design methodology, system overview, and design considerations. This section also includes detailed figures that augment the description of the LPS critical design.

2.1 Design Methodology

This section briefly describes the methods used to perform the design, including the design process and products.

2.1.1 Design Process

The LPS detailed design has been developed using *SEAS System Development Methodology* (SSDM) tailored to suit the LPS project environment. The LPS design has been accomplished by performing the following major activities:

- Development of an LPS top-level architecture that is based on LPS structured design, conforms to the selected hardware configuration and constraints, and maximizes the use of commercial off-the-shelf (COTS) items in its design.
- Design of the LPS database that is based on the refinement of a logical model of LPS data.
- Design of a user interface for the LPS based on software requirements and operations concepts.
- Design of the LPS subsystems using a computer-aided software engineering (CASE) tool—Cadre/Teamwork—which supports the structured design methodology.
- Identification of LPS issues that, when resolved, may impact the LPS critical design.

2.1.2 Design Products

Several products are produced as a result of the detailed design phase of the LPS. They include

- A model of the LPS design in Cadre/Teamwork that describes the software design; this model includes
  - Structure charts—Graphical representations of the hierarchy of the modules; structure charts consists of modules, data, and control couples passed between modules
  - Module specifications (M-specs)—Describe what function a module is to perform and how a module performs its function
  - Data dictionary—Provides definitions for data items in the LPS software design
  - Entity relationship diagrams (ERDs)—Define the attributes of each entity and depict the relationship among entities; also includes constraints that enforce the integrity of the database
- This ABS

- An interface definition document (IDD) that defines the interface among LPS subsystems
- An LPS users guide
- An LPS user interface prototype that supports operational scenario and allows developers and users in the Earth Resources Observation Systems (EROS) Data Center (EDC) to explore the LPS operations together, better understand requirements, and implement improvements quickly and efficiently.

2.1.3 Design Conventions

The LPS design is expressed in a notation adapted from E. Yourdon and L. Constantine and supported by CADRE/Teamwork. The notation is illustrated in Figure 2–1. The notation is consistent with SSDM Standard 4205, but includes the following enhancements and exceptions:

- Unit representations on structure charts include only a title; they do not include a purpose statement.
- The conditional execution symbol also is used for transaction centers.
- Signals (UNIX software interrupts) are represented by asynchronous invocation from an off-sheet connector without source.
- Special notations are used for LPS global functions, data units, and database access routines as illustrated in Figure 2–1g.

2.2 LPS System Overview

The LPS is a major component of the Landsat 7 Ground System, and is located at the EDC in Sioux Falls, South Dakota, along with the Landsat Ground Station (LGS) and the EDC Distributed Active Archive Center (DAAC). The LGS captures Enhanced Thematic Mapper Plus (ETM+) wideband data directly from the Landsat 7 spacecraft using a 10-meter antenna and two 150-Mbps X-band return links, separating each X-band data into two 75-Mbps I and Q channels, and transmitting the acquired wideband data over four 75-Mbps LGS output channels to the LPS. The LGS provides a fifth 75-Mbps spare output channel to be used as a backup for its four primary channels.

The LPS coordinates its operations with the LGS in accordance with the Landsat 7 contact period schedules to receive the return-link wideband data in real time from all four output channels of the LGS into its four wideband data stores, one per LPS string. Each LPS string retrieves the received wideband data from its wideband data store [a 32-gigabyte redundant array of independent disks (RAID)] and processes it to Level 0R format at a rate equal to or greater than 7.5 Mbps. The Level 0R processing generates browse, metadata files, and band files (1-6L or 6H-8) and stores the files on the LPS back-end data store (32-gigabyte RAID) for staging and data transfer to the EDC DAAC. The LPS also provides a fifth spare string to be used as backup for its four primary strings.

The LPS receives return-link wideband data from the LGS on a Landsat 7 contact period basis and generates Level 0R band and metadata files on a subinterval basis and the browse image files

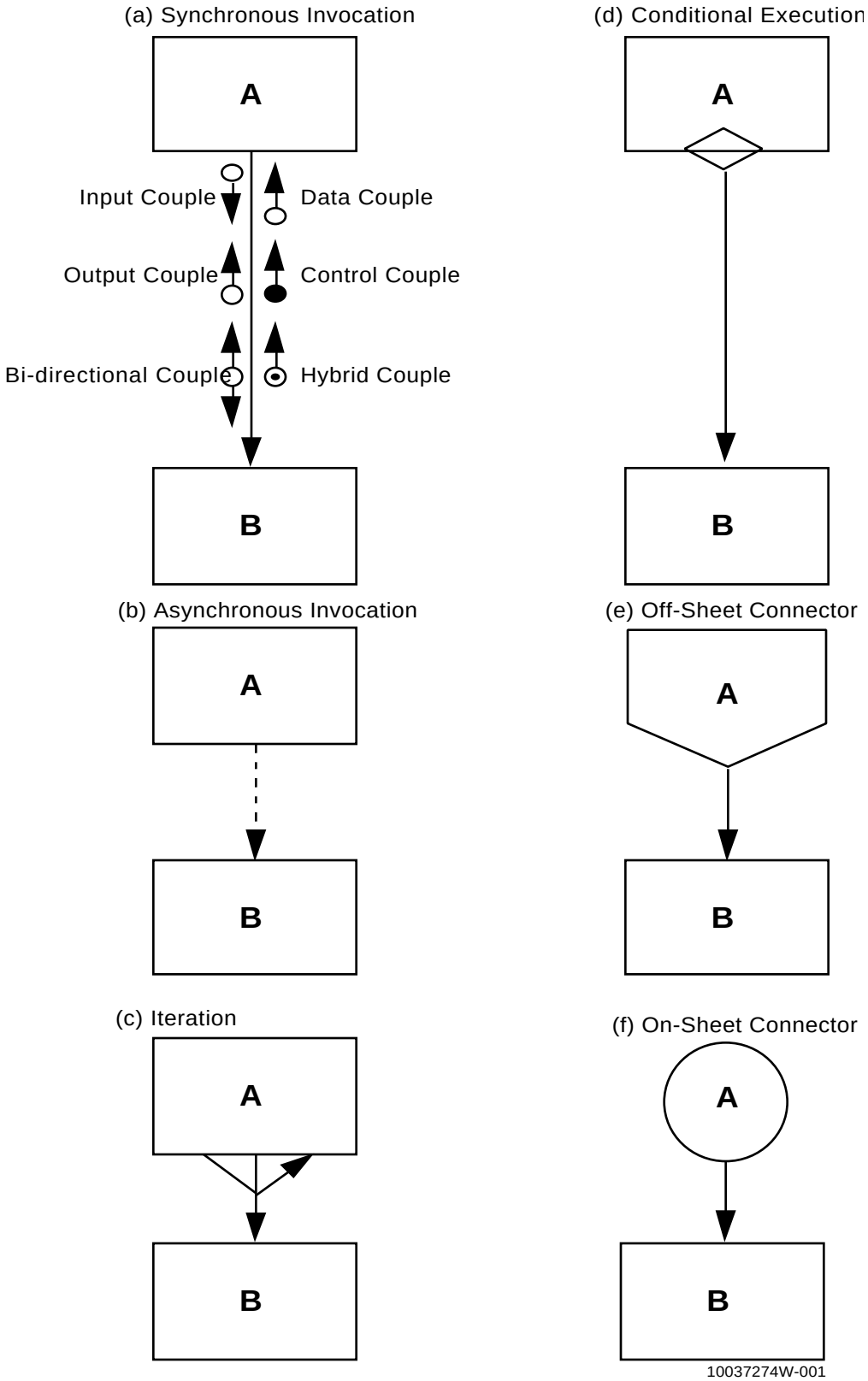


Figure 2–1. LPS Design Conventions (1 of 2)

(g) Unit Types

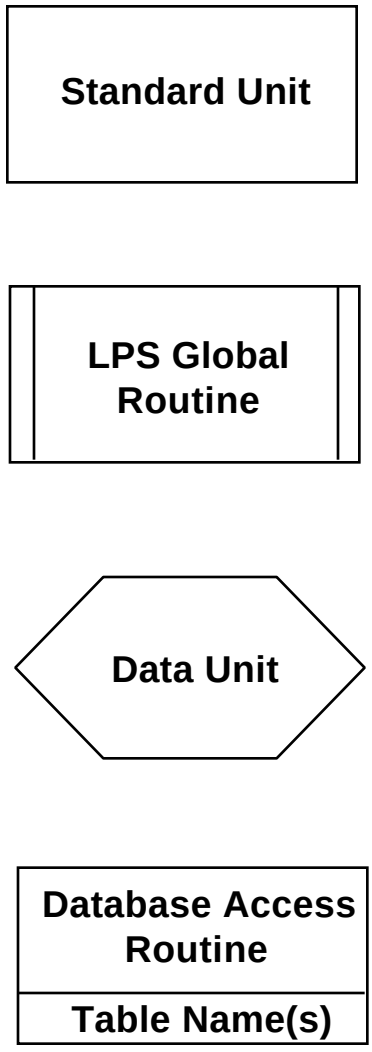


Figure 2–1. LPS Design Conventions (2 of 2)

on a scene basis. The LPS completes Level 0R processing of the wideband data, archiving the raw wideband data to 30-day storage, and notification of data availability to the EDC DAAC within 16 hours of its receipt. The LPS is sized to receive, process, and deliver to the EDC DAAC the equivalent of 250 ETM+ scenes of wideband data from the LGS and from Alaska on a daily basis. In addition to the 250 ETM+ scenes, the LPS can reprocess the equivalent of 25 ETM+ scenes of wideband data on a daily basis. The LPS maintains the overall processing throughput performance as long as the raw wideband data received from the LGS meets the bit error rate (BER) of 1 bit error in 10^5 bits.

The LPS also generates return-link data quality and accounting (Q&A) information from the wideband data received on a Landsat 7 contact period basis. The LPS provides Level 0R Q&A information as part of the metadata to the EDC DAAC for each subinterval processed from each Landsat 7 contact period.

The LPS consists of five logically independent processing strings. Four strings are used to support normal operations, while the fifth string is the backup, test, development, and training string. Figure 2–2 illustrates the LPS functional block diagram.

2.2.1 Hardware

The hardware that comprises the LPS is primarily five Silicon Graphics, Inc. (SGI) Challenge XLs. Each XL houses eight central processing units (CPUs), 512 megabytes of random access memory (RAM), one 4.3-gigabyte system disk, one 4mm digital audio tape (DAT) tape drive, one 8mm tape, a CD-ROM, and two IO4 boards providing the interface to the SGI storage devices. Externally resides two 34-gigabyte disk arrays, one Digital Linear Tape (DLT™) library, and one label printer. The SGI XLs provide data capture, Level 0R processing, and the staging area for subsequent transfer to the EDC DAAC.

The operational configuration also includes two Xterms, two Indy workstations, and two Hewlett-Packard (HP) laser printers. The Xterms facilitate the commanding of the LPS software on each string, and the Indy workstations provide a display for the moving window displays (MWDs) and console terminals for each XL via serial lines. The LPS software is designed such that it may be run remotely from any X Windows based terminal.

The Xterms, workstations, and XLs are connected to one another through an Ethernet network that permits a flexible environment for the operators and system maintenance personnel, and some limited resource sharing among the systems. The SGI XLs are equipped with fiber-distributed data interface (FDDI) devices connecting the LPS to the EDC DAAC for high-speed data transfers. Figure 2–3 illustrates the operational network configuration for the LPS relative to the LGS and EDC DAAC.

2.2.2 Software

LPS software context level 0 diagram (Figure 2–4) shows the interactions of the LPS software subsystems and which ones interface with the external ground system components.

The scope and allocation of requirements to each subsystem were driven by the LPS requirements from the F&PS. These requirements have been further divided between operations,

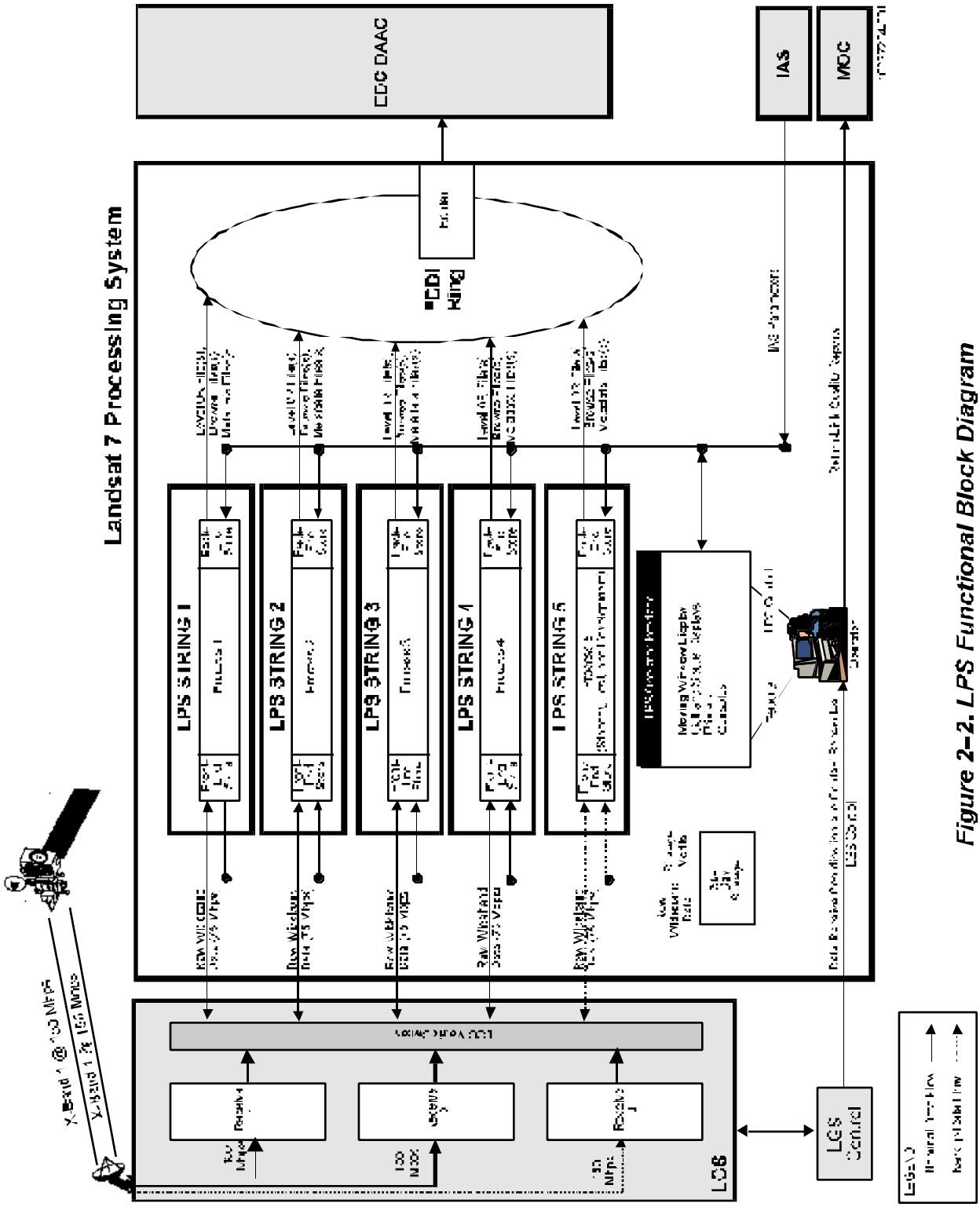


Figure 2–2. LPS Functional Block Diagram

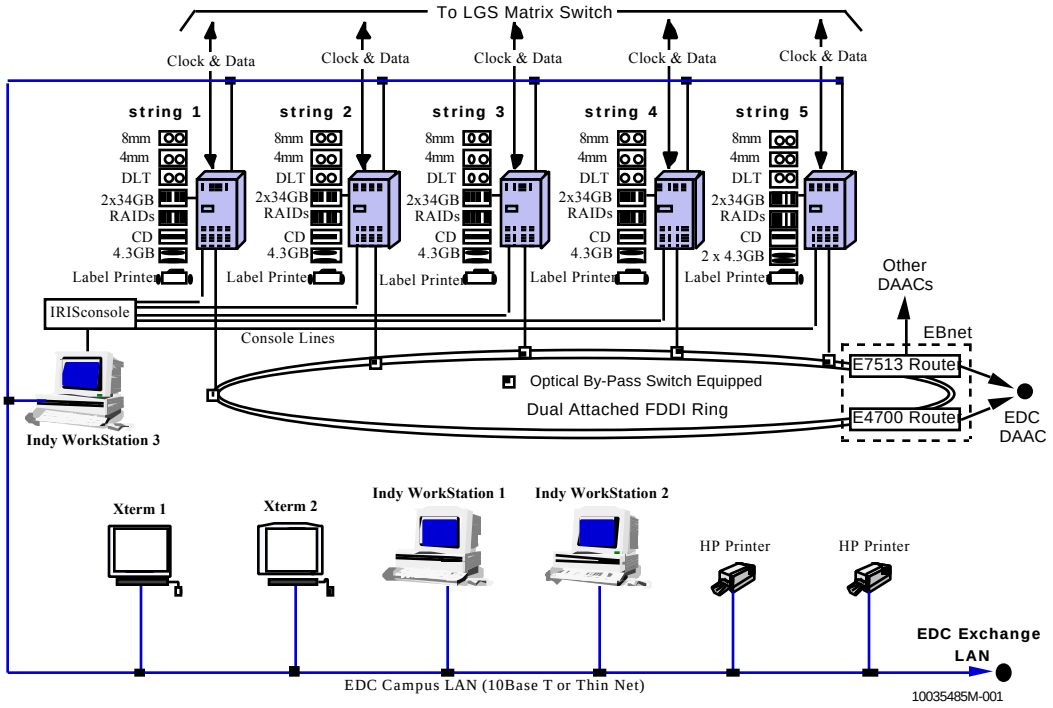


Figure 2–3. LPS Operational Network Configuration

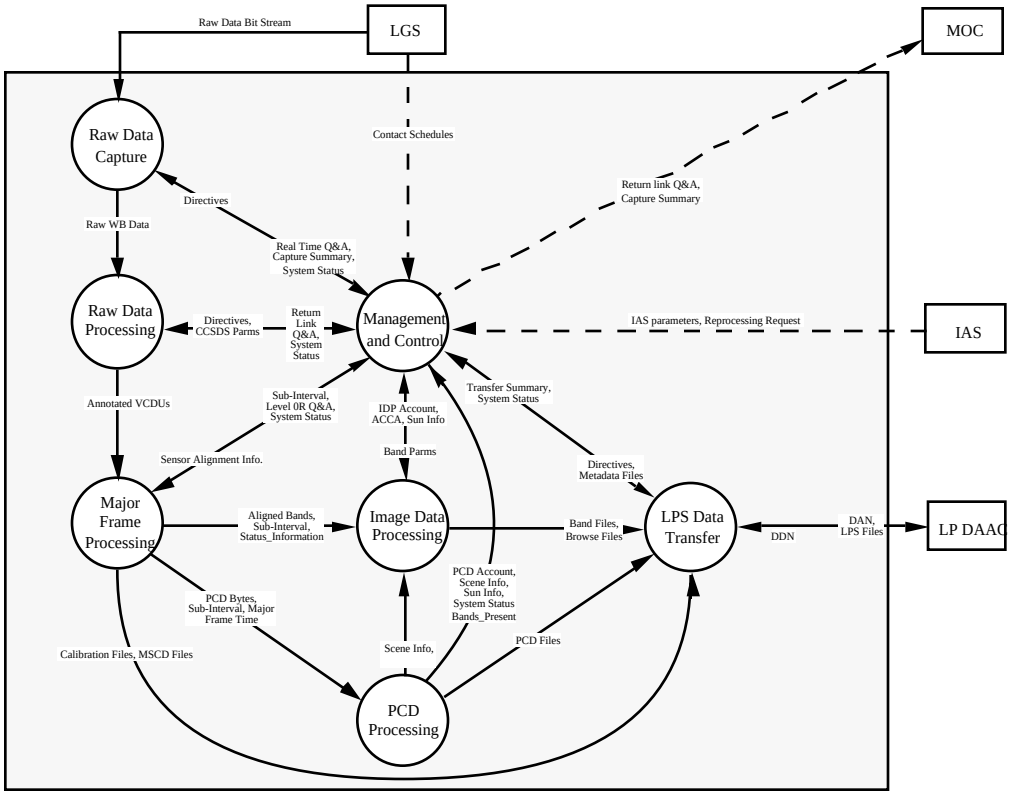


Figure 2–4. LPS Software Context Level 0 Diagram

hardware, and software and analyzed to form this ABS. Several requirements apply to all LPS subsystems (refer to the LPS SDS, Appendix A, "Requirements Assigned all Subsystems" section).

The raw data flows into the raw data capture subsystem (RDCS), where it is captured to the LPS front-end RAID. The raw data is then sent to the raw data processing subsystem (RDPS), where all transmission artifacts are removed, leaving spacecraft mission data. This mission data (Ann_VCDU) is then passed to the major frame processing subsystem (MFPS), which interprets the mission data and passes payload correction data (PCD) to the PCD processing subsystem (PCDS) and image data to the image data processing subsystem (IDPS). The MACS accepts Q&A information from the processing subsystems and creates a metadata file. When the metadata file is created, the LPS data transfer subsystem (LDTS) notifies the EDC DAAC that the output files are ready for transfer. EDC DAAC is then responsible for transferring the files from the LPS. Once EDC DAAC has successfully transferred the files, the LDTS deletes them from LPS back-end storage device.

2.2.3 External Interfaces

The LPS interfaces with five external elements of the Landsat 7 Ground Processing Segment: Mission Management Office (MMO), LGS, EDC DAAC, Mission Operations Center (MOC), and Image Assessment System (IAS). See Figure 2–5.

The MMO defines the day-to-day guidelines, rules, and priorities governing the processing of data by the LPS.

EDC DAAC electronically receives a data availability notification (DAN) containing the LPS file list for a processed contact period once all wideband data from a Landsat 7 contact period is processed. EDC DAAC transfers the Level 0R band, browse image, and metadata files (LPS files) from the LPS output store and electronically acknowledges receipt of the data to the LPS.

The MOC receives the wideband data summary report within 5 minutes of data receipt for each contact to confirm the loss data or the receipt of poor quality data. This report may be sent via fax or phone.

The IAS delivers parameters files to the LPS that specify the calibration, radiometric correction, and geometric correction values to be used in processing of Landsat 7 data. The parameter file is received electronically via File Transfer Protocol (FTP) from the IAS. This file is expected once every 3 months and will be approximately 1 megabyte in size.

2.3 Design Considerations

This section presents the concepts and philosophy that drive the design.

2.3.1 LPS Top-Level Architecture Control Strategies

This subsection defines the rationale behind LPS top-level control strategy design. There are two significant areas of the design: raw data capture, handled by the RDCS, and L0R processing, handled by the IDPS, MACS, MFPS, RDPS, and LDTS.

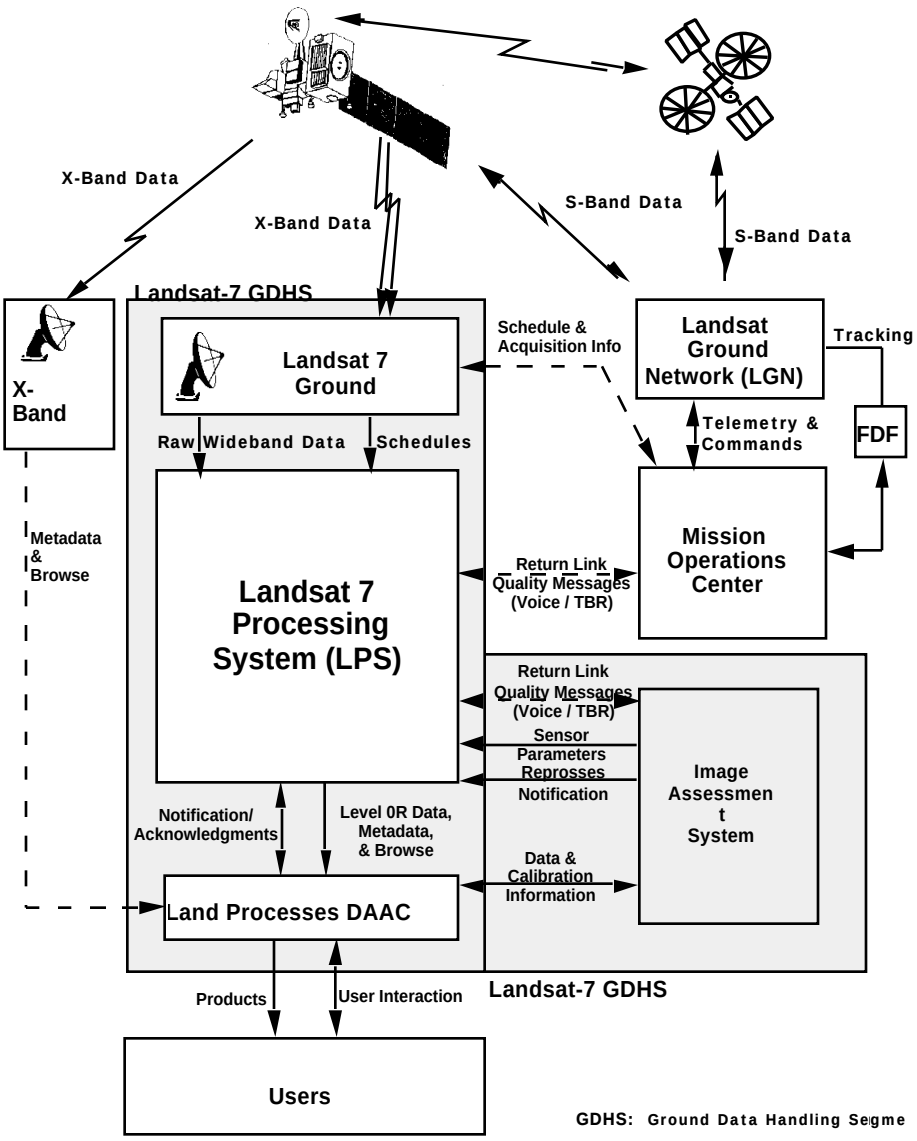


Figure 2–5. LPS External Interfaces

For raw data capture handling, two related questions were posed and considered:

1. Whether RDCS is able to capture data when the LPS database is down
2. Whether RDCS allows both manual and scheduled capture

A cost/benefit analysis was done. The determination was to provide both of these capabilities.

For Level 0R processing, a process in the MACS will be invoked by operator command to initiate Level 0R processing on a specified contact. The MACS process will invoke RDPS, MFPS, IDPS, PCDS, and LDTS programs; monitor their progress; and exit only when all have terminated. Each subsystem program will accept command line arguments as appropriate and return a status value via exit(). The MACS process will announce successful completion only when all subsystems involved have exited with success.

A uniform command line argument format is desirable. An option/value format (e.g., -f thefile.out) will be used for all command line arguments.

Environment variables and database tables are both used as mechanisms to store filename generation parameters, output file locations, and other runtime parameters.

Two options for suspending Level 0R processing during data capture were discussed:

1. Each Level 0R process becomes a member of a single group. When capture begins, an RDCS process will send a suspend signal to all members of the group. When capture ends, the RDCS will send a resume signal.

2. A single mutual exclusion mechanism (semaphore, lock file, or other appropriate method) on the capture disk is designed such that RDPS will perform a read() on the capture disk only when the mutual exclusion mechanism indicates that capture is not going on. When capture is to begin, the RDCS sets the mutual exclusion mechanism so that the RDPS will wait until capture has completed before doing the next read.

The second option is simpler in that it involves fewer delivered source instructions (DSI) in fewer subsystems. It may be more efficient because it allows processing on data already retrieved from the capture disk. However, the mechanism is effective only if the capture disk is the only resource such that contention for the resource impacts data capture. Because other resources are required for data capture, the former method was selected.

The LPS design places the function of acquiring and deallocating shared resources for Level 0R processing, such as first-in-first-out (FIFO), semaphores, and shared memory, within the top-level Level 0R process. The top-level process will not continue unless all shared resources are successfully allocated. Other Level 0R processes need only attach to the shared resources. The design eliminates difficulties arising from the indeterminate order of execution of concurrent processes. All concurrent Level 0R processes may attach to shared resources identically because the resource is guaranteed to be created before the process begins execution.

2.3.2 Global Library Areas

A library of general purpose functions was defined during preliminary design. During detailed design, the scope of the library was expanded and its functionality was refined.

2.3.3 Database Interface

A trade-off study resulted in the decision to place all database access through embedded Structured Query Language (SQL) in separate database access routines (DBARs) that contain little more than the SQL statements themselves. The rationale for doing so was that these smaller units run less risk of encountering compile-time problems with the ORACLE-embedded SQL preprocessor.

2.3.4 Error Handling Philosophy

The LPS error handling philosophy distinguishes several classes of errors. A primary distinction is between correctable and uncorrectable errors. The LPS philosophy for correctable errors is to report and handle the error in the unit that detects it.

The LPS design has two broad classes of correctable errors. The first and more important class are those that occur during raw data capture. Because raw data capture failure means that captured data

is lost, the LPS design for raw data capture attempts to continue under most circumstances, possibly with degraded operation. The second class of correctable errors occurs when error-reporting thresholds for Level 0R processing cannot be retrieved from the database. Correction consists of using default values.

Within the class of uncorrectable errors, the LPS error handling philosophy distinguishes fatal and non-fatal errors. Fatal errors cause a process to terminate abnormally and immediately. No opportunity for recovery exists at the time of the error's occurrence. Catastrophic hardware failures or receipt of a UNIX terminate signal are events that can cause a process to terminate abnormally. Non-fatal errors make further processing impossible, but do not abort the process. The process is able to respond to the error at the time of its occurrence.

For fatal errors, the LPS error-handling philosophy provides for error detection by a monitor process. In general, each process that performs a function is invoked and monitored by a controlling process. A fatal error in the functioning process is detected by the monitoring process.

In most cases, correction after fatal errors is unnecessary; the LPS's state remains unchanged until a process terminates successfully. An important exception occurs with Level 0R processing. Level 0R processing involves the concurrent execution of eight processes. During execution, output files are produced incrementally and database inserts are committed. The abnormal termination of any single process requires explicit operations to delete output files and delete the inserted database records to restore the LPS to a consistent state. The Level 0R top-level processes perform these operations.

With the exception of raw data capture errors and threshold fetch errors, the LPS design treats most errors as uncorrectable and terminates execution gracefully when the error occurs. This decision is intended to minimize development costs without impact to system requirements. The decision is feasible because, with the exception of raw data capture, operations on the LPS are repeatable. An uncorrectable error that terminates execution is therefore corrected by repeating the operation that terminated.

Robust error reporting is central to the LPS error-handling philosophy for all types of errors. Robust error reporting consists of detailed error and status messages delivered throughout processing.

LPS error reporting uses the IRIX operating system's system log services to record all status and error message from the LPS. Each error and status message is time-tagged and assigned a severity value by the unit that generates the error. To aid in troubleshooting, each message also contains the name of the unit that generates the error and the source code line number of the error report function call.

The LPS design reports each detected error by the unit that first encounters the error. Any calling unit that changes its flow of control because of the error also reports the change. This action results in a chain of error messages at different levels of abstraction, indicating both the particular event in error and the more global changes in execution flow resulting from the error.

The chain of error messages is valuable for troubleshooting, but can be problematic for operators who are flooded with a barrage of messages for a single error. To provide both a valuable troubleshooting tool and a useful operational environment, the LPS design logs all error messages

to a journal file. The operator monitors the LPS by viewing addition to this file in real time. The LPS design allows the operator to specify a severity level cut-off that the LPS will use to filter the messages displayed. Only messages with the specified, or higher, severity will be displayed.

2.3.5 File Maintenance

The LPS design outputs the following file types:

- Raw data capture files
- Level 0R output files
- Trouble files containing unprocessable data from the RDPS, MFPS, and PCDS
- Files containing DANs generated for transmission to the EDC DAAC
- Temporary files created by various LPS software components and deleted at program termination

The LPS design treats the management of these file types differently.

For raw data capture files, the LPS design assumes that the operator explicitly deletes raw capture files after they have been copied to removable media and processed to Level 0R. The LPS design includes utilities to reduce the probability that files are not inadvertently deleted before having been copied to removable media and to maintain consistency between capture files online and in the LPS database.

For Level 0R output files, the LPS design automatically deletes files successfully transferred to EDC DAAC (subject to operator override capabilities). The LPS design automatically deletes output files generated during a Level 0R processing run that did not completely successfully execute.

The LPS design does not allow trouble files to grow indefinitely large. When files reach a tunable threshold value, no further writes to the file occur. Trouble files and DAN files are never deleted automatically, but will be purged as part of normal system maintenance.

Temporary files are normally deleted by the subsystem on exit, whether with successful completion or an error. In the case of catastrophic errors, some temporary files may remain. These files will be purged as an operational step in system recovery. Instructions for purging such files appear in the LPS Users Guide.

Section 3. Operational Scenarios

This section discusses the LPS operational scenarios and includes scenarios for both automatic and manual processing.

3.1 Introduction

LPS operational scenarios represent sequences of activities performed by operations personnel as they relate to LPS software. The scenarios may be divided into three categories: system setup, normal operations, and contingency operations. These categories represent the majority of the activities performed by LPS operations personnel.

- System setup – Activities involved in LPS initialization and configuration
  - Start up LPS string.
  - Receive contact schedule from the LGS.
  - Set up LPS strings for data capture.
  - Execute LPS functions test and product verification.
  - Receive parameters from the IAS.
  - Adjust LPS Level 0R parameters.
  - Adjust LPS Level 0R thresholds.
- Normal operations – Activities performed routinely to accomplish Landsat 7 data processing within the LPS
  - Receive data from the LGS (automatic).
  - Process data to Level 0R.
  - Monitor data quality.
  - Transfer files to the EDC DAAC.
  - Monitor LPS-EDC DAAC file transfers.
  - Restage data for reprocessing.
  - Support operational training and test.
  - Generate reports.
- Contingency operations – Activities performed in response to abnormal conditions
  - Receive data from the LGS (manual with database operating).
  - Receive data from the LGS (manual without database).
  - Respond to failure in LGS-to-LPS interface.
  - Respond to failure in LPS-to-EDC-DAAC interface.

– Respond to exhaustion of LPS output storage capacity.

– Respond to LPS string failure.

– Restore LPS string.

The operational scenarios are strongly affected by the LPS architecture presented in Figure 3–1. This architecture includes five logically independent processing strings. LPS operations use four processing strings at all times to support normal operations. The fifth string is available for LPS test and maintenance support, as required, as a backup string for the four operational strings and as a site for operations training and testing. The architecture also includes two workstations that serve as the consoles for the four operational strings and the backup string, with each displaying up to two MWDs on command.

The methods to accomplish operations involving all four operational strings (e.g., concurrent loading of calibration parameters) are detailed in the LPS Users Guide.

The sequence of steps in each operational scenario is described in the following subsections. Each description includes an annotation that designates the LPS subsystem or external entity performing the step. A functional flow block diagram illustrating the sequence of steps accompanies most scenarios. Rounded rectangles enclose the distinct steps in the scenario. Rounded rectangles with dotted borders indicate optional steps. Squared rectangles indicate a collection of steps detailed in other operational scenarios. Vectors indicate flow control and establish the sequence of steps. A background grid indicates the entity that performs each step. Note that because the LPS has five independent strings (four operational and one back up), some input to the LPS must be repeated once for each string. The number of repetitions is indicated by a number appearing next to a vector.

LPS operational scenarios are based on the following assumptions:

· The management and control subsystem (MACS) schedules data capture for a single, i.e., the next, contact period. Details of the MACS design for scheduling data capture are presented in Section 10.

· The MACS starts up the LDTS data delivery notice (DDN) receiving process on system startup.

· The LPS operator can change parameters or thresholds during Level 0R processing, but they will not take effect until the next time Level 0R processing is invoked.

3.2 System Setup

System setup operations scenarios describe the sequences of operator activities involved in LPS initialization and configuration.

3.2.1 Start Up LPS String

The LPS operator brings an LPS string up from a completely powered-off state by following these steps (shown in Figure 3–2):

1. LPS Operator—Power on/boot operations interface workstation and LPS string.

2. LPS Operator—Log onto operations interface workstation.

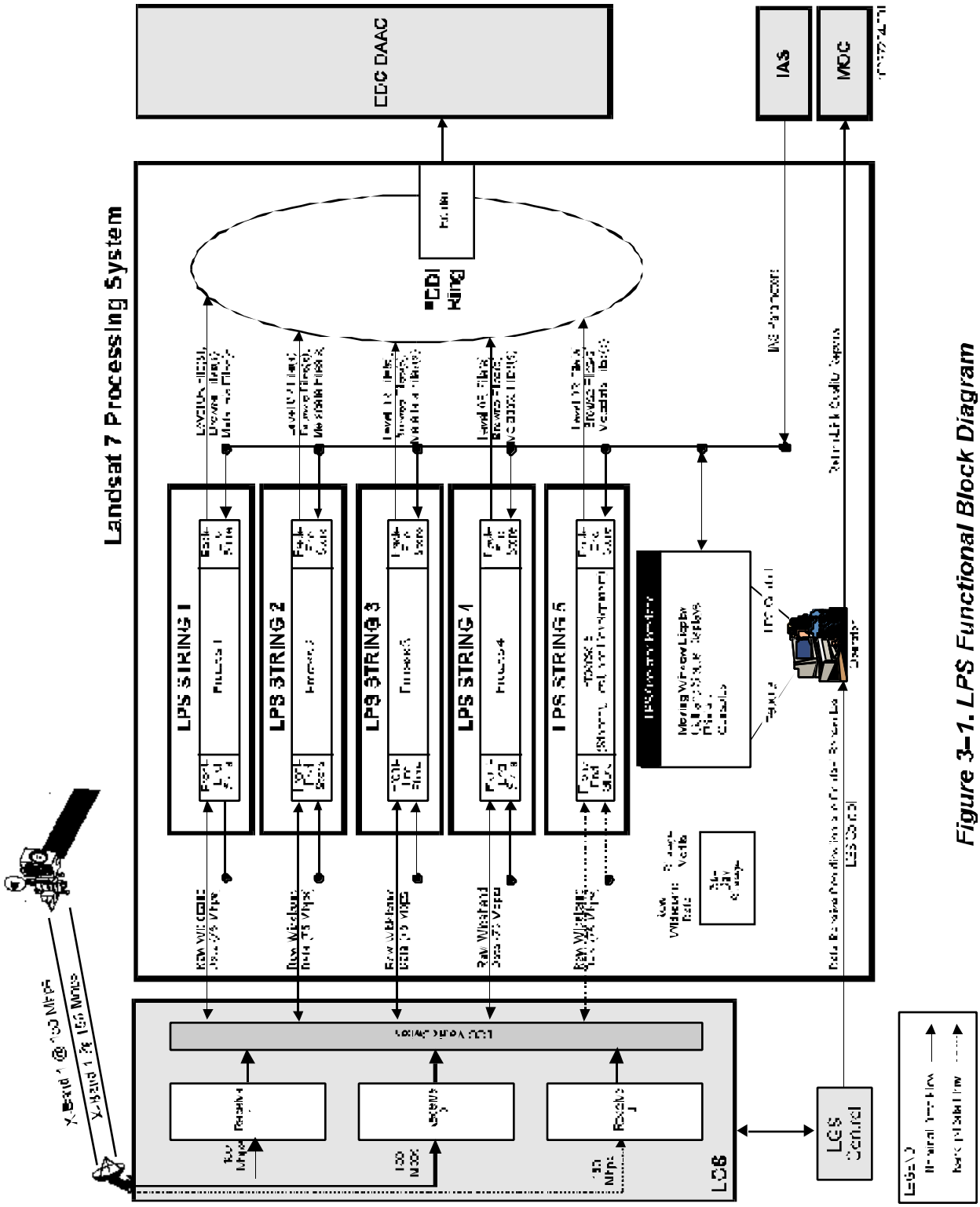


Figure 3–1. LPS Functional Block Diagram

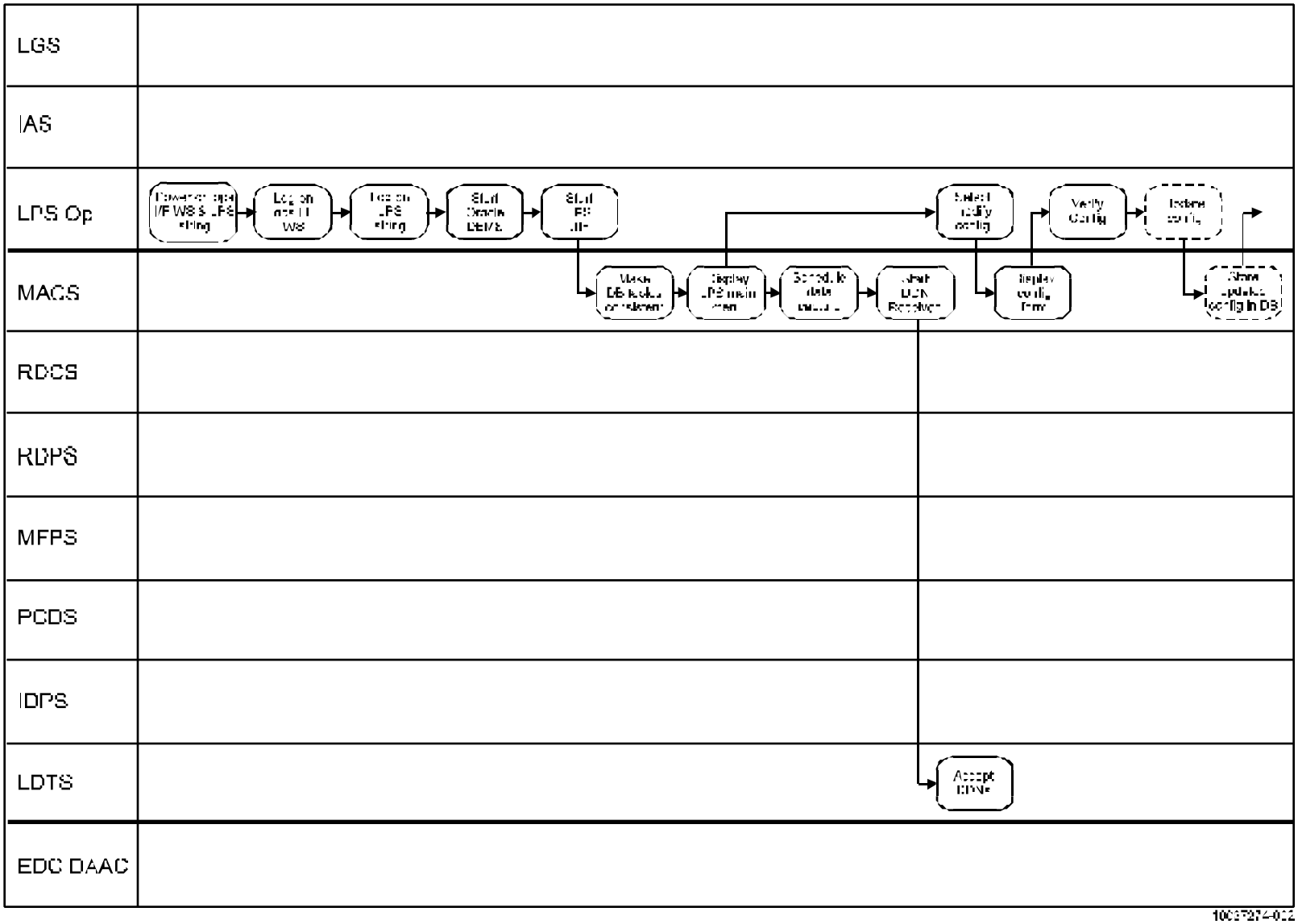


Figure 3–2. Start Up LPS String (1 of 2)

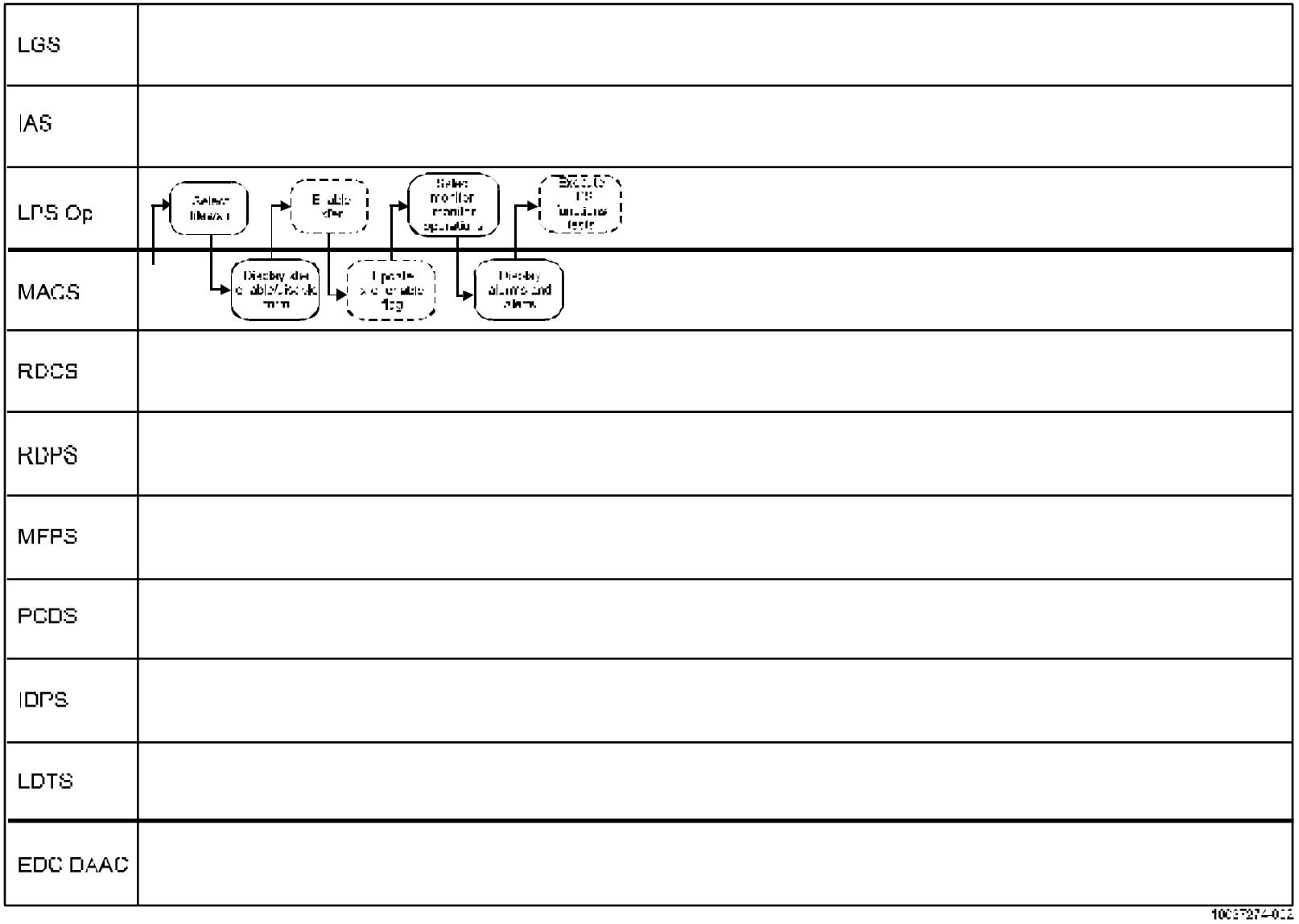


Figure 3–2. Startup LPS String (2 of 2)

3. LPS Operator—Log onto LPS string console.

4. LPS Operator—Start up ORACLE database management system (DBMS) on the LPS string (if not part of system boot).

5. LPS Operator—Telnet/rlogin from operations interface workstation to LPS string.

6. LPS Operator—Start LPS user interface.

7a. MACS—Make database tables consistent.

7b. MACS—Display LPS main menu.

7c. MACS—Schedule automatic capture for next scheduled contact.

7d. MACS—Start DDN receiver process.

8. LPS Operator—Select modify/configuration menu options.

9. MACS—Display configuration form.

10. LPS Operator—Verify LPS string configuration (hardware string identifier, LGS channel identifier, spacecraft identifier, instrument identifier).

11. LPS Operator (optional)—Update LPS string configuration.

12. MACS (optional)—Validate and store updated configuration in the LPS database.

13. LPS Operator (Optional)—Select files/enable disable transfer menu options.

14. MACS (Optional)—Display LPS file transfer enable/disable form.

15. LPS Operator (Optional)—Enable file transfer to EDC DAAC, if required.

16. MACS—Update file transfer enable flag in the LPS database.

17. LPS Operator (Optional)—Execute LPS functions test and product verification.

18. LPS Operator—Select monitor/monitor operations menu options to display LPS alarms and alerts.

3.2.2 Receive Contact Schedule From the LGS

The LPS operator receives a hardcopy contact schedule from the LGS operator. The LPS operator inputs the schedule to the LPS software on each LPS string. (A method for concurrently updating all four strings is TBD.) The steps performed (shown in Figure 3–3), once for each operational string, are as follows:

1a. LGS—Send contact schedule to LPS operator.

1b. LPS Operator—Receive contact schedule.

2. LPS Operator—Select setup/modify schedule menu options.

3. MACS—Display schedule modification form.

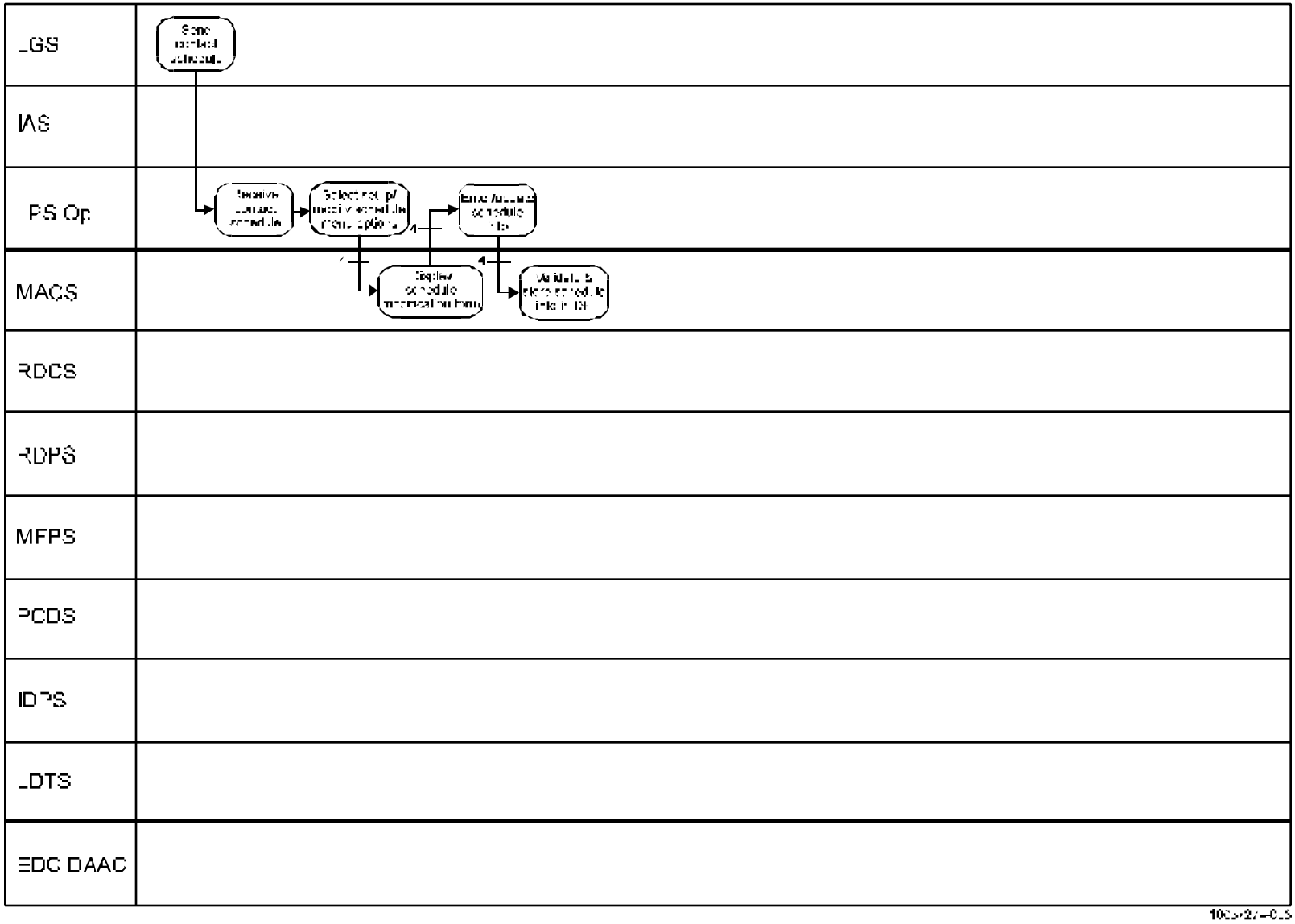


Figure 3–3. Receive Contact Schedule From the LGS

4. LPS Operator—Update scheduled contact entries for contacts already in LPS database but modified on new schedule. Enter new contact entries for contacts not in LPS database.

5. MACS—Validate and store modified contact schedule entries in LPS database.

3.2.3 Set Up LPS Strings for Data Capture

The LPS operator verifies that the LPS configuration is correct and functional prior to each contact period. The steps performed for LPS data capture set up (shown in Figure 3–4) are as follows (a method for concurrently updating all four strings is TBD):

1. LGS (Optional)—Switch LGS channels to new configuration.

2. LPS Operator—Review LGS contact schedule for new configuration.

3. LPS Operator (optional) —If LGS channel reconfiguration is required, select setup/configuration menu options

4. MACS (optional)—Display configuration form.

5. LPS Operator (optional)—Update LGS channel for new configuration.

6. MACS (optional)—Validate and store updated LGS channel in the LPS database.

7. LPS Operator—Check available disk space for system data stores using IRIX utilities and clean up as necessary.

8. LPS Operator—Check available disk space on capture disk using IRIX utilities. It is an operational decision whether existing files will be deleted or data will not be captured when insufficient disk space is available.

9. LPS Operator—Execute LPS functions test for LPS functions and product verification.

3.2.4 Execute LPS Functions Test and Product Verification

The LPS operator verifies that the LPS configuration is correct and functional prior to each contact period. The steps performed for LPS functions test and product verification (shown in Figure 3–5) are as follows:

1. LPS Operator—Select files/disable transfer menu options to disable automatic DAN transmission to EDC DAAC on string to be tested.

2. MACS—Set file transfer flag in LPS database to “disabled.”

3. LPS Operator—On the string to be tested, issue command to process disk-resident test data to Level 0R.

4. RDPS, MFPS, PCDS, IDPS—Process data to Level 0R. (See Section 3.3.2 for steps to perform Level 0R processing.)

5. LPS Operator—Verify correct operation by examining moving window display (MWD), output files, and LPS Q&A report. (See Section 3.3.8 for the steps to generate this report.)

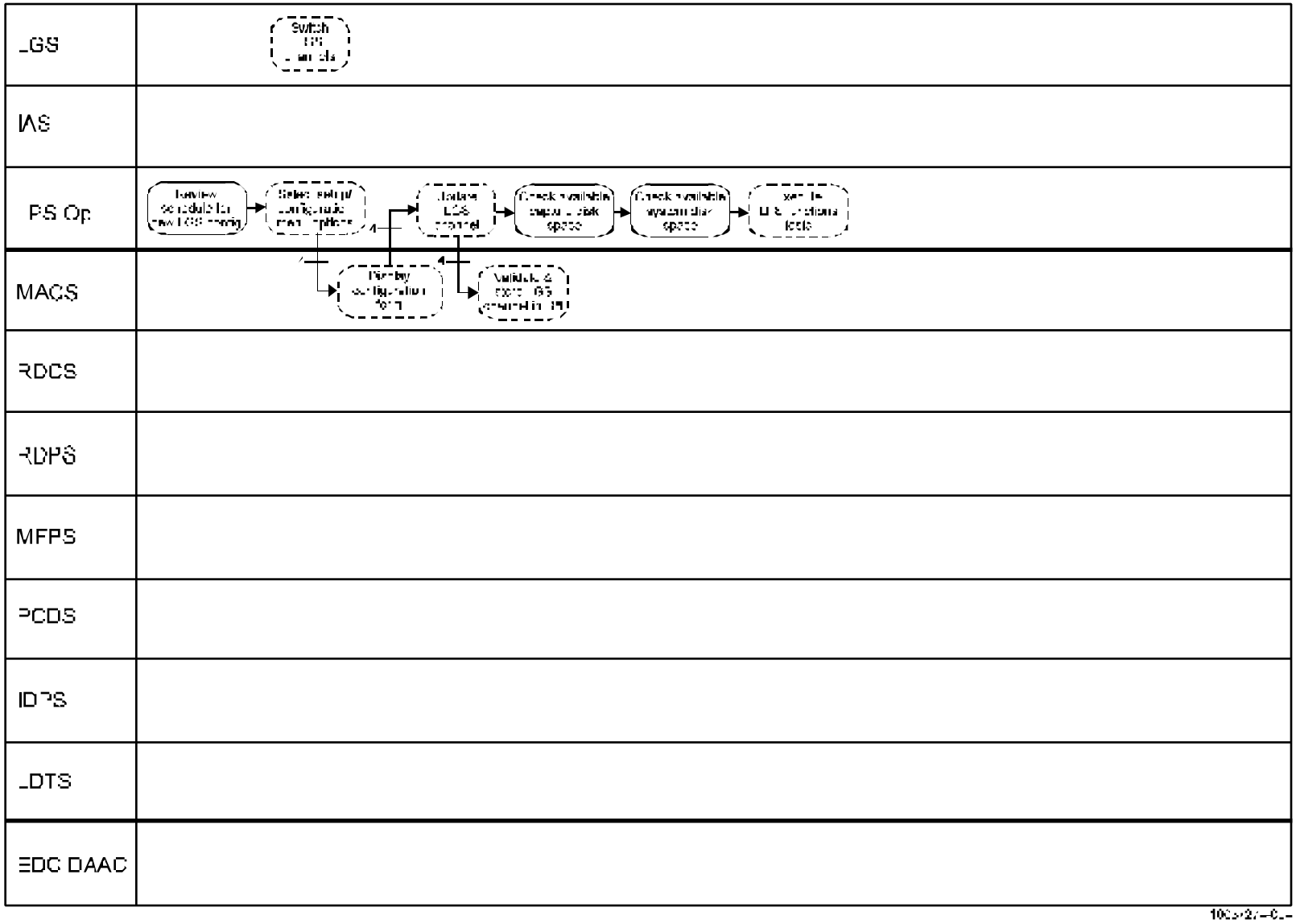


Figure 3–4. Set Up LPS Strings for Data Capture

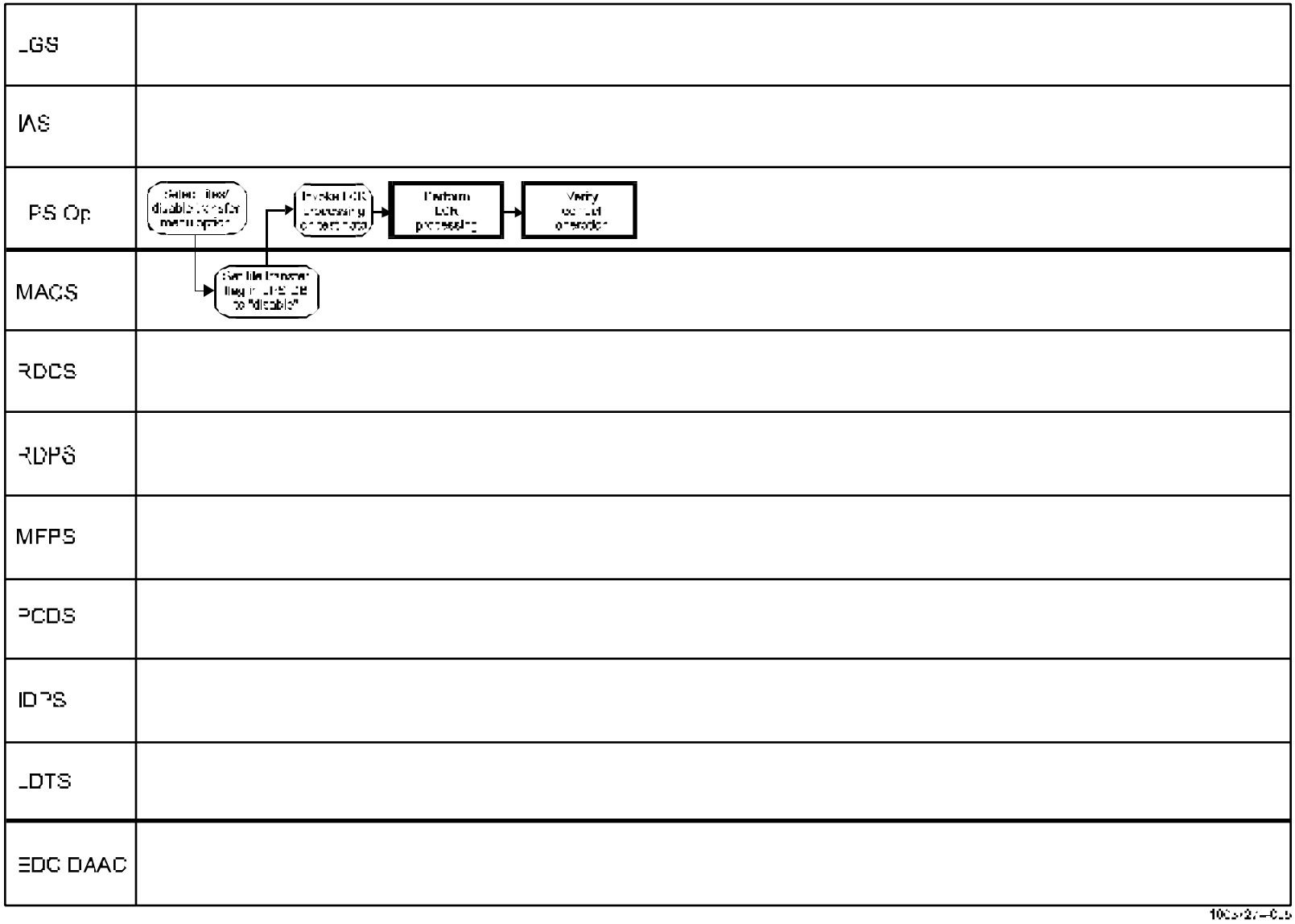


Figure 3–5. Execute LPS Functions Test and Product Verification

3.2.5 Receive Parameters From the IAS

IAS uses FTP to send the file of calibration parameters. The LPS operator inputs the parameters to the LPS software on the LPS test string and tests for 1 week. At the end of the week, if the results on the test string are acceptable, the parameters are promoted to the operational strings. (A method for concurrently updating all four strings is TBD.) The steps performed when a file of calibration parameters is received (shown in Figure 3–6) are as follows:

1. IAS—FTP new parameters file to LPS.

On the test string:

2. LPS Operator—Select “Ingest IAS Parms” from the SETUP drop-down menu list.
3. MACS—Display a default ingest IAS filename.
4. LPS Operator—Update filename as appropriate and press “OK.”
5. MACS—Validate and store new parameters in LPS database.
6. LPS Operator—Test for 1 week with actual LPS data.

On the operational strings:

7. LPS Operator—Promote parameters to the operational strings.

3.2.6 Adjust LPS Level 0R Parameters

The LPS operator adjusts LPS Level 0R parameters whenever it is determined that Level 0R output can be increased by adjusting the processing parameters. The steps performed to adjust Level 0R parameters (shown in Figure 3–7) are as follows:

1. LPS Operator—Determine desired parameter values.
2. LPS Operator—Select setup/RDP parameters menu options on the LPS test string.
3. MACS—Display RDP parameters form.
4. LPS Operator—Update RDP parameters.
5. MACS—Validate and store RDP parameters in LPS database.
6. LPS Operator—Select MFP parameters option on LPS test string.
7. MACS—Display MFP parameters form.
8. LPS Operator—Update MFP parameters.
9. MACS—Validate and store MFP parameters in LPS database.
10. LPS Operator—Select IDP parameters option on LPS test string.
11. MACS—Display IDP parameters form.
12. LPS Operator—Update IDP parameters
13. MACS—Validate and store IDP parameters in LPS database.

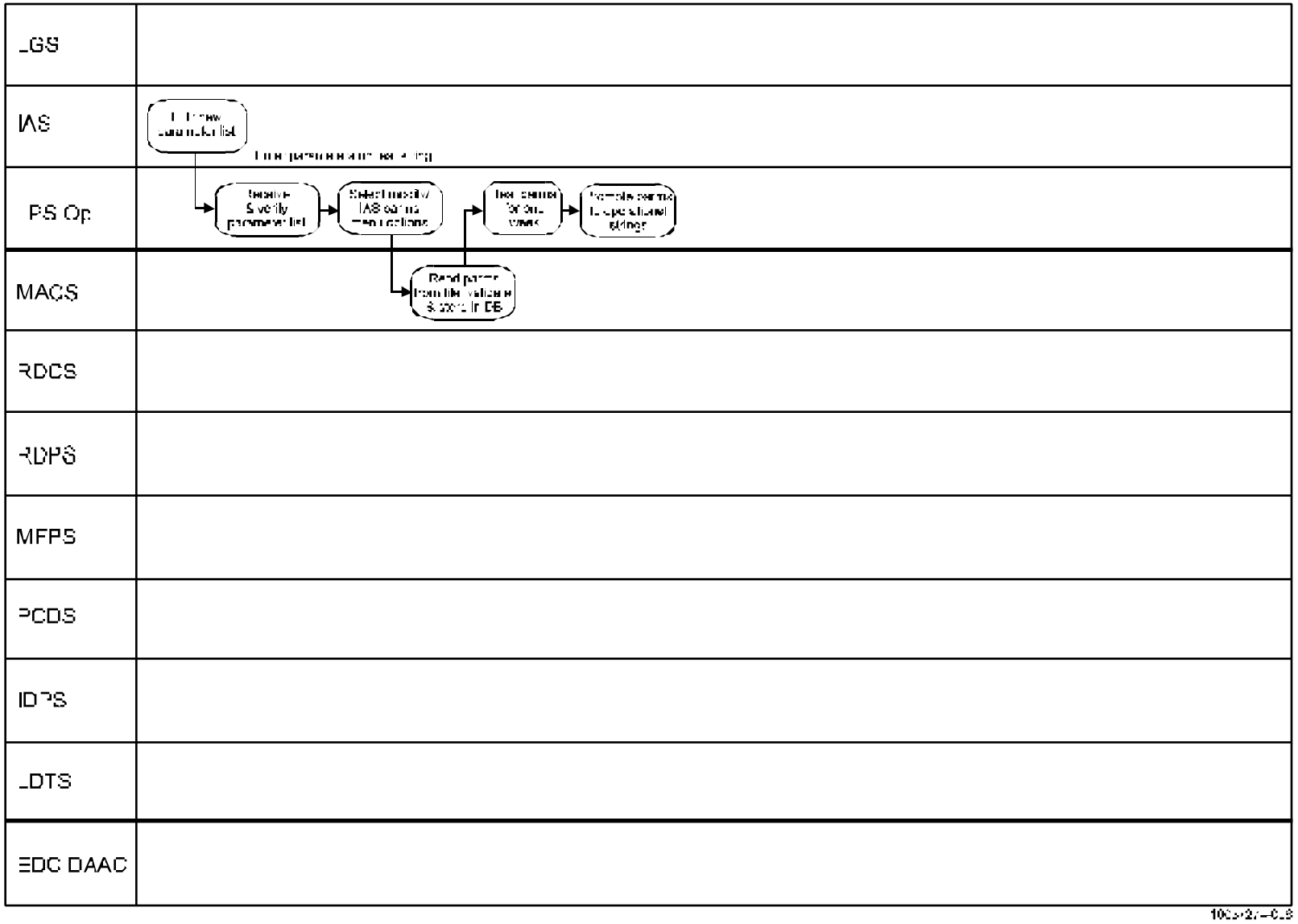


Figure 3–6. Receive Parameters From the IAS

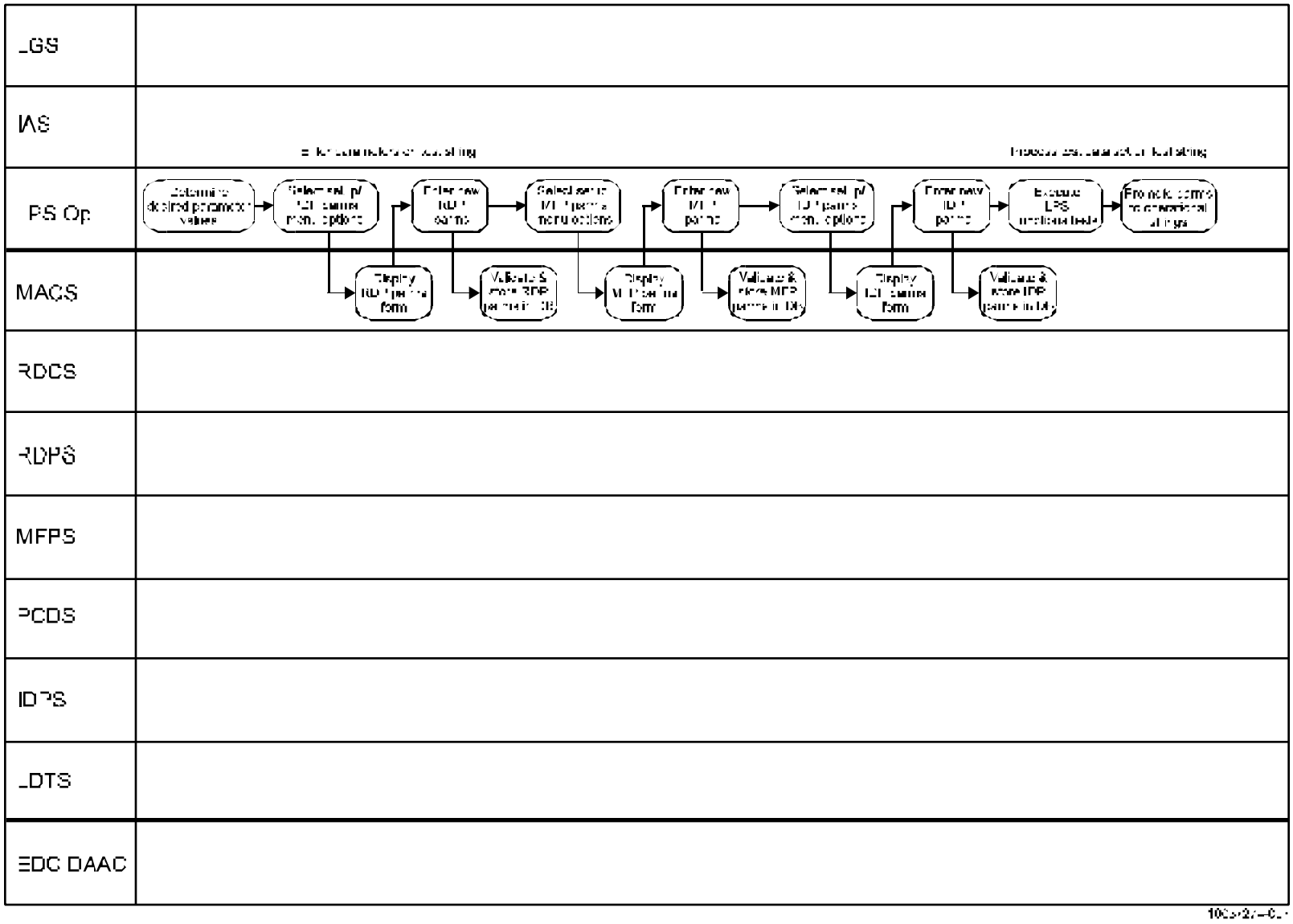


Figure 3–7. Adjust LPS Level 0R Parameters

3.2.7 Adjust LPS Level 0R Thresholds

The LPS operator can adjust LPS Level 0R error reporting thresholds whenever excessive noise in raw wideband data causes a proliferation of alarms and alerts. The steps performed to adjust Level 0R thresholds (shown in Figure 3–8) are as follows:

1. LPS Operator—Determine desired threshold values.
2. LPS Operator—Select set up/modify thresholds menu options.
3. MACS—Display thresholds form.
4. LPS Operator—Update thresholds.
5. MACS—Validate and store updated thresholds in the LPS database.

3.3 Normal Operations

Normal LPS operations scenarios describe the sequences of operator activities performed routinely to accomplish Landsat 7 data processing within the LPS.

3.3.1 Receive Data From the LGS (Automatic)

LPS raw data capture normally is performed automatically based on the contact schedule input by the LPS operator. Each string automatically captures the LGS output channel to which it is configured. This scenario presupposes the successful execution of the "Set Up LPS Strings for Data Capture" scenario described in Section 3.2.3. The steps performed to receive data automatically from the LGS (shown in Figure 3–9) are as follows:

1. MACS—Retrieve next contact from contact schedule and schedule data capture start.
2. MACS—At a number-of-seconds parameter set by the operator prior to scheduled data start time, invoke RDCS with a scheduled stop time.
3. RDCS—Begin data capture.

4a. RDCS—Every 5 seconds, log the number of 10-MB buffers that have been received.

4b. MACS—Display periodic captured data volume reports.

4c. LPS Operator—Monitor data capture.

5a. RDCS—At the scheduled stop time, cease data capture.

5b. RDCS—Insert data receive summary record into LPS database.

6. LPS Operator—Select reports/data receive summary menu options, specify this contact and select display or print options.
7. MACS—Invoke data receive summary report generator.
8. RDCS—Generate data receive summary.
9. MACS—Display/print data receive summary.
10. LPS Operator—Provide the MOC with data receive summary using voice link or fax.

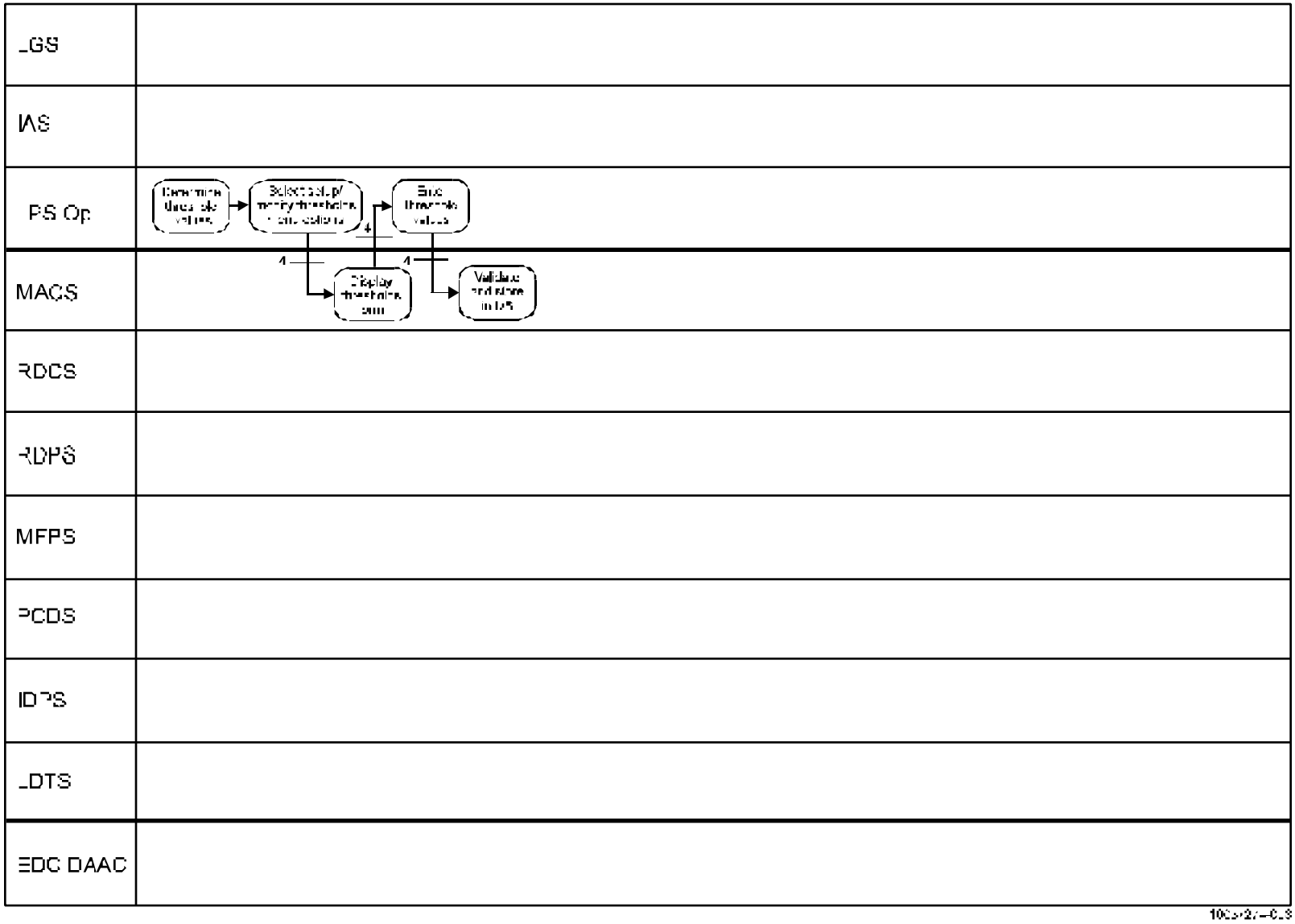


Figure 3–8. Adjust LPS Level 0R Thresholds

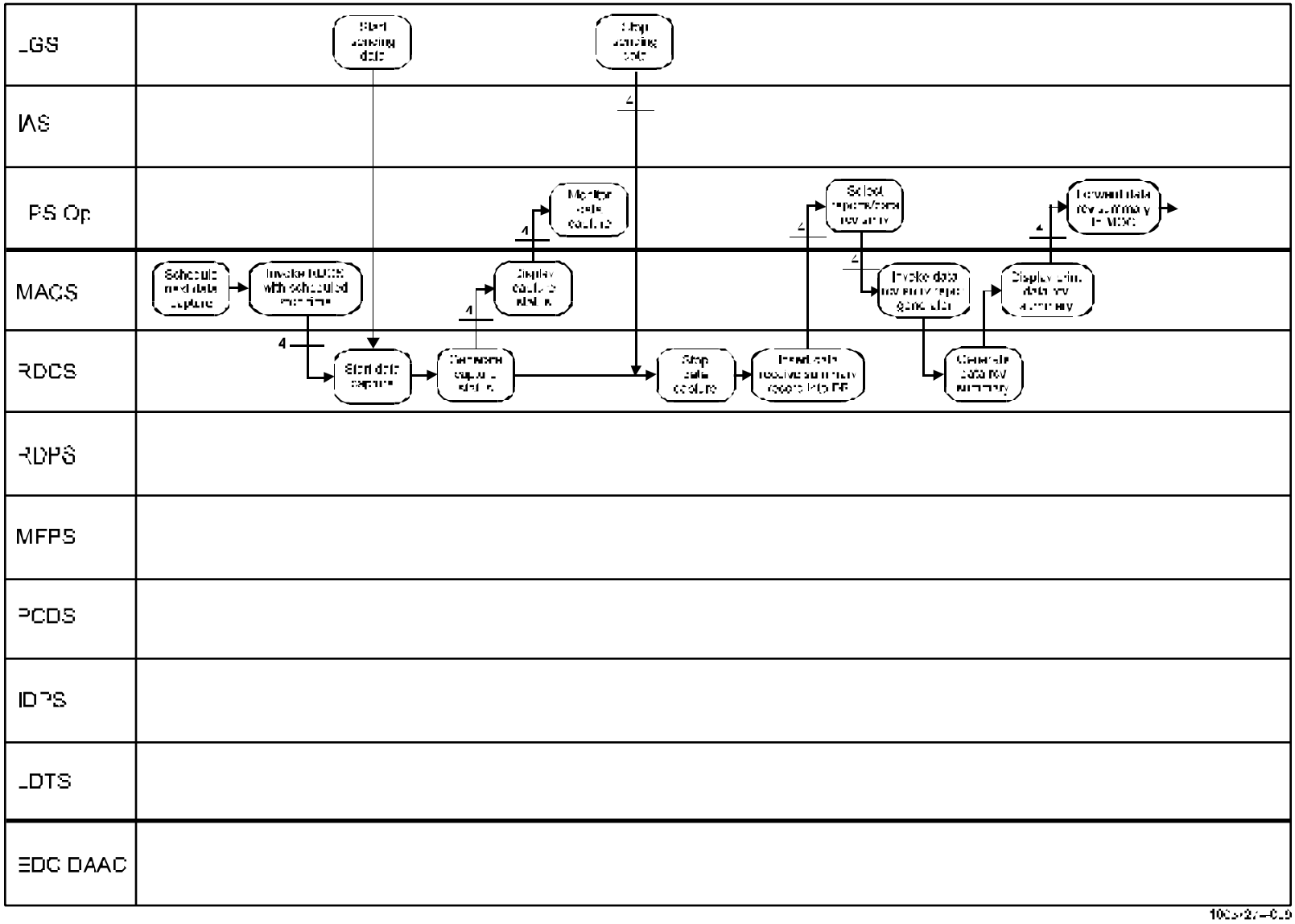


Figure 3–9. Receive Data From the LGS (1 of 2)

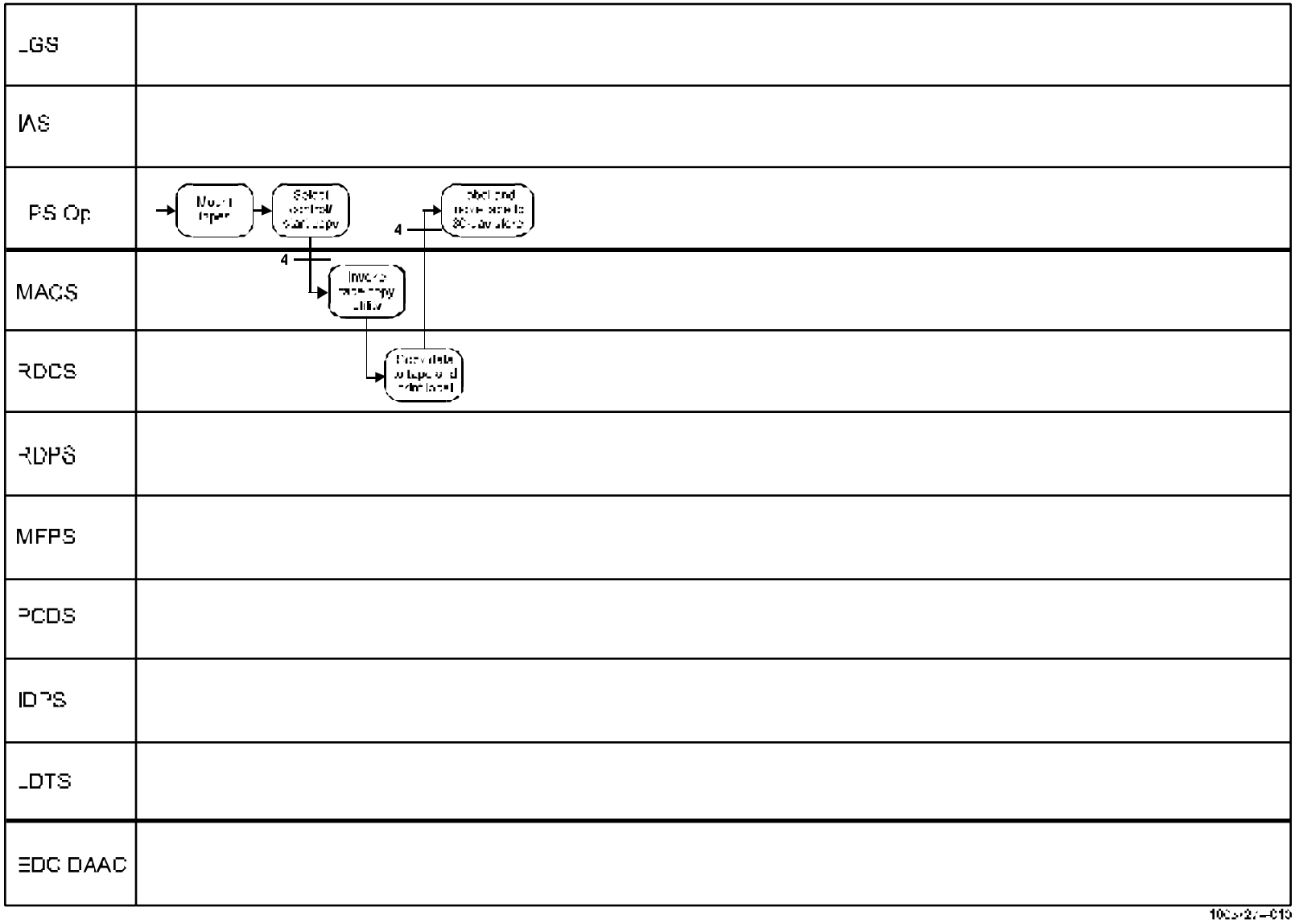


Figure 3–9. Receive Data From the LGS (2 of 2)

11. LPS Operator—Mount tape.

12. LPS Operator—Select control/start copy menu options, specifying tape drive and contact period.

13. MACS—Send start copy command.

14. RDCS—Copy specified contact period to tape and print label.

15. LPS Operator—Apply label and move tape to 30-day storage.

3.3.2 Process Data to Level 0R

The LPS operator initiates the processing to Level 0R of raw wideband data on each LPS string that stores data for the contact period. Optionally, the operator uses the MWD on each string to display (with reduced pixel size, but not reduced resolution) images of the Landsat 7 ETM+ science instrument data for visual verification and cursory quality assessment before its transfer to EDC DAAC. At the end of processing, the operator optionally generates Q&A reports. The steps performed to process data to Level 0R (shown in Figure 3–10) are as follows:

1. LPS Operator—Verify that sufficient disk space is available in the data transfer store for output products using IRIX system commands.

2. LPS Operator—Select control/start processing menu options, selecting contact period to be processed.

3. MACS—Invoke Level 0R processing programs.

4a. RDPS—Process specified contact period.

4b. MFPS—Process specified contact period.

4c. PCDS—Process specified contact period.

4d. IDPS—Process specified contact period.

5a. EDC Operator (optional)—For each string, select MWD option, data format (format 1 or format 2), bands to be viewed, and display device.

5b. MACS—Validate MWD parameters and store in database

5c. MACS—Initialize window.

5d. IDPS—Generate MWD image data on a continual basis.

5e. IDPS—Load generated images to workstation.

5f. EDC Operator—Periodically examine image data for obvious problems.

5g. EDC Operator—If image data is inferior, inform shift manager.

6. MACS—Generate metadata for processed contact period.

7. MACS—Invoke LDTS to send DAN for this contact period.

8a. LDTS—Open connection to EDC DAAC and send DAN to EDC DAAC.

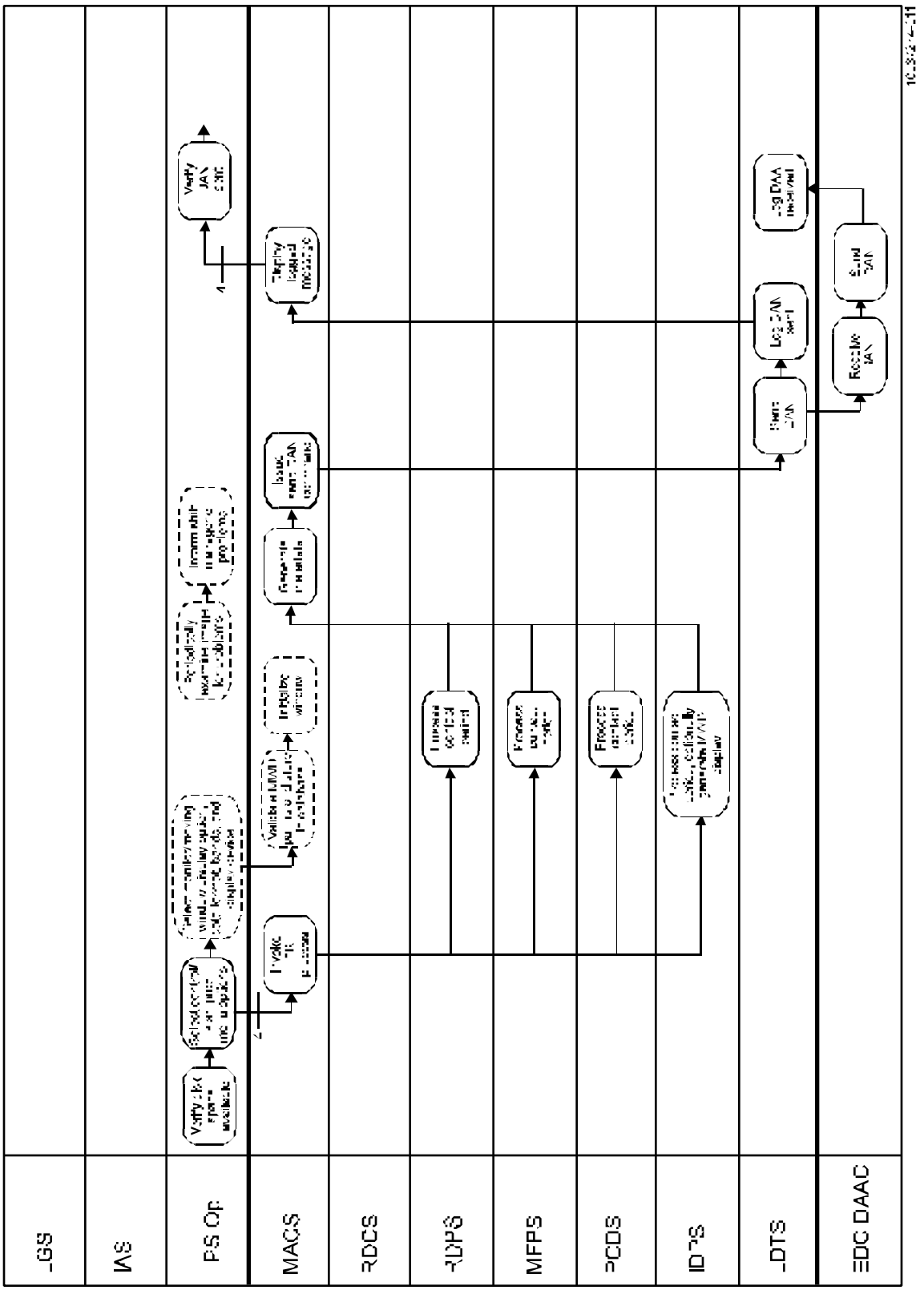


Figure 3–10. Process Data to Level 0R (1 of 2)

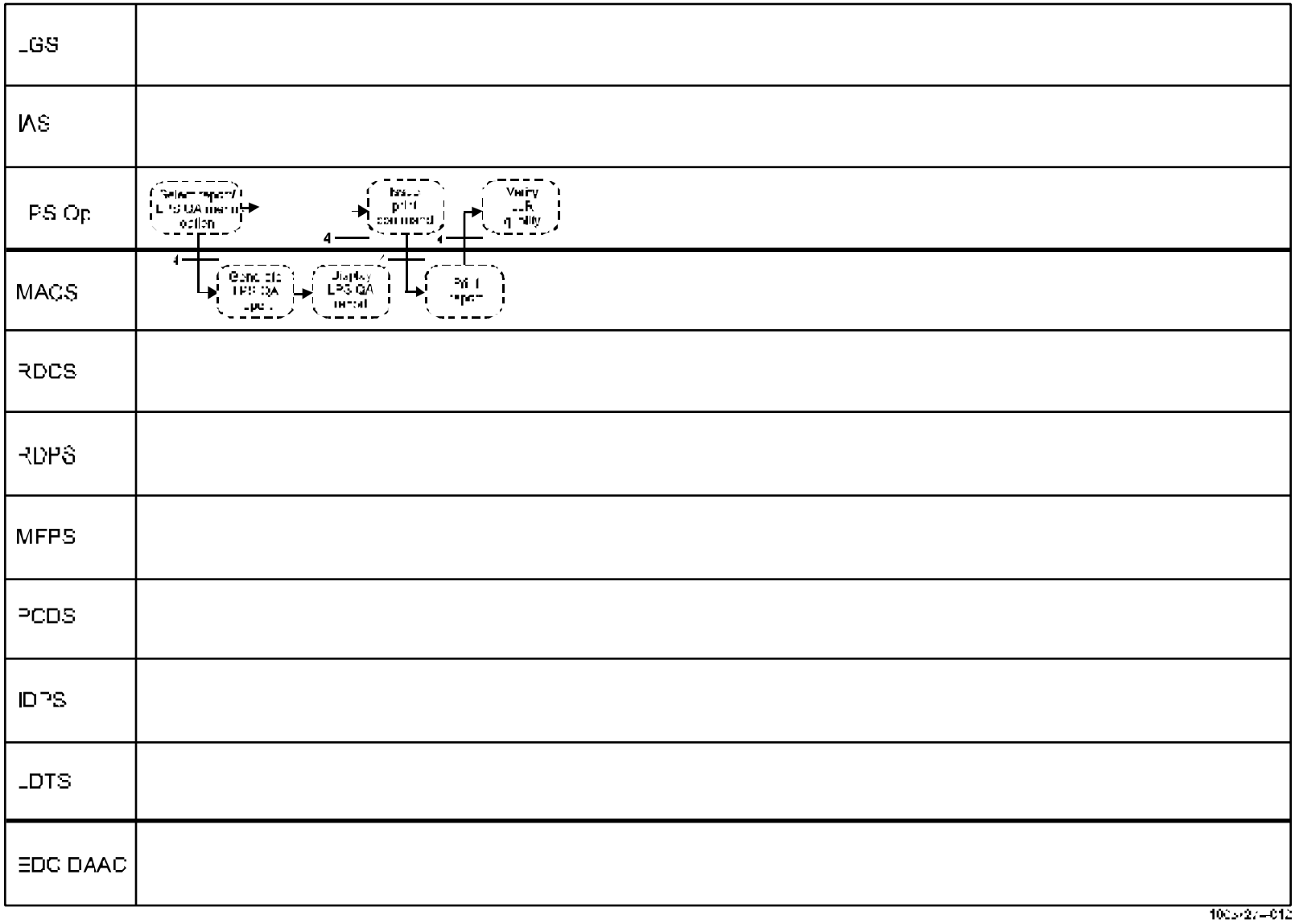


Figure 3–10. Process Data to Level 0R (2 of 2)

8b. EDC DAAC—Receive DAN and respond with data availability acknowledgment (DAA).

9a. LDTS—Log to LPS journal that DAN was sent and acknowledged.

9b. LDTS—Terminate connection.

9c. MACS—Display "DAN sent" message (assumes operator has LPS monitor operating).

9d. LPS Operator—Verify DAN was acknowledged by EDC DAAC.

10. LPS Operator (optional)—Select LPS Q&A report, specifying this contact period, and display/print option.

11. MACS (optional)—Invoke report generator and print report.

3.3.3 Transfer Files to EDC DAAC

Output files are transferred automatically from the LPS to EDC DAAC. (This scenario assumes a DAN has already been sent.) This section describes the automatic transfer scenario. The steps performed to transfer data from the LPS to EDC DAAC (shown in Figure 3–11) are as follows:

1. EDC DAAC—Open an FTP connection to the LPS string containing the desired file(s).

2. LDTS—Authenticate and accept connection (it is assumed that this step is performed by the IRIX FTP daemon or a similar system program).

3. EDC DAAC—Request transfer of desired file(s).

4. LDTS—Transmit desired file(s) to host (it is assumed that this step is performed by the IRIX FTP daemon or a similar system program).

5. EDC DAAC—Terminate the FTP connection.

6a. EDC DAAC—Open a Transmission Control Protocol (TCP) connection to the LPS string from which files were obtained.

6b. EDC DAAC—Send the DDN.

7. LDTS—Respond with a data delivery acknowledgment (DDA).

8. EDC DAAC—Terminate TCP connection.

9. LDTS—Delete files successfully transferred.

10. LDTS—Update file transfer records in the database.

3.3.4 Monitor LPS-to-EDC DAAC File Transfers

Output files are transferred automatically to EDC DAAC. The LPS operator monitors the LPS-to-EDC DAAC interface. The steps performed by the operator to monitor LPS-to-EDC DAAC file transfers (shown in Figure 3–12) are as follows:

1. LPS Operator—Monitor interface with EDC DAAC via LPS log and IRIX system utilities. For example, all FTP sessions and/or each file transfer can be logged by command arguments to ftp'd. Current connections can be monitored by Netstat.

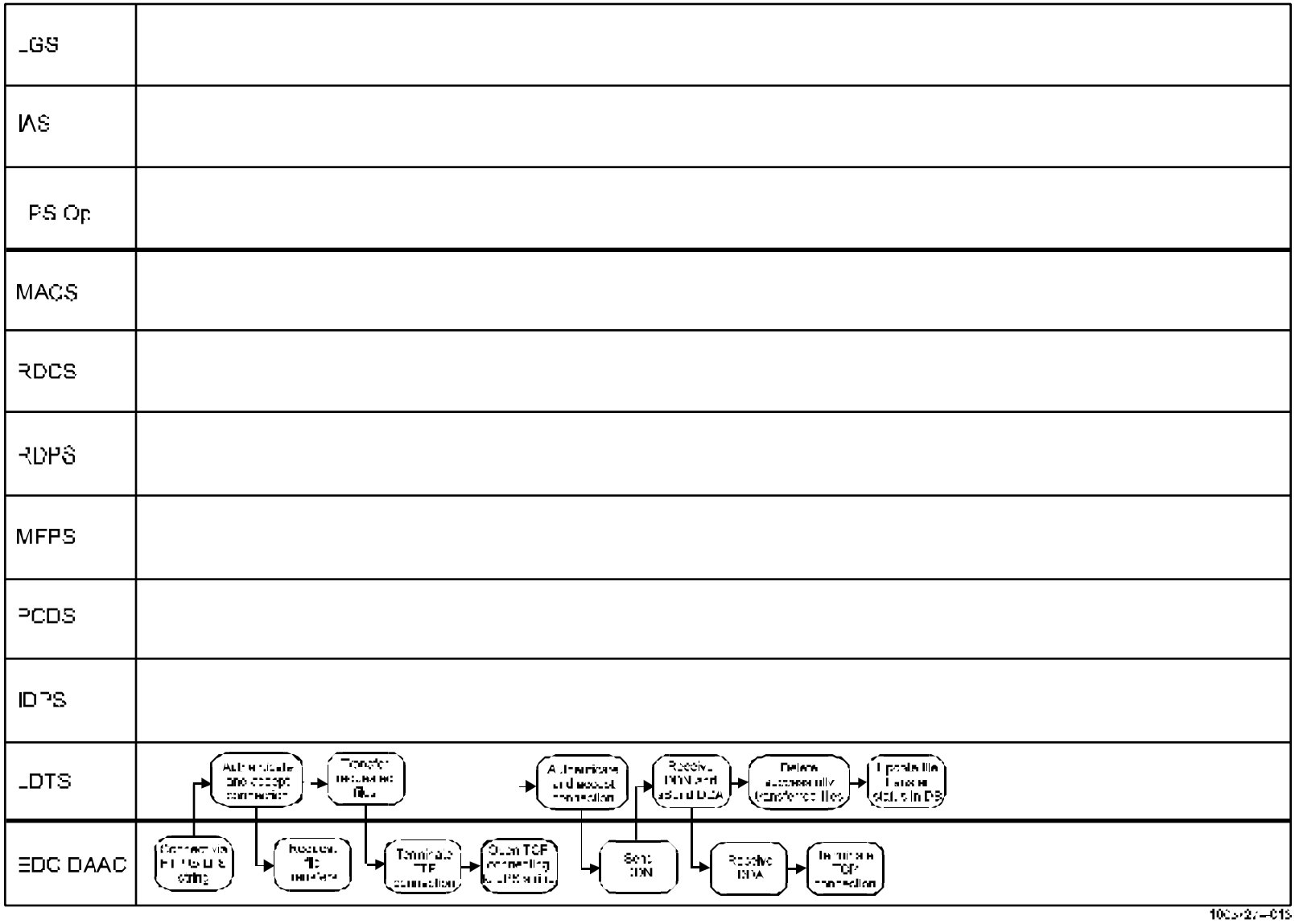


Figure 3–11. Transfer Files to EDC DAAC

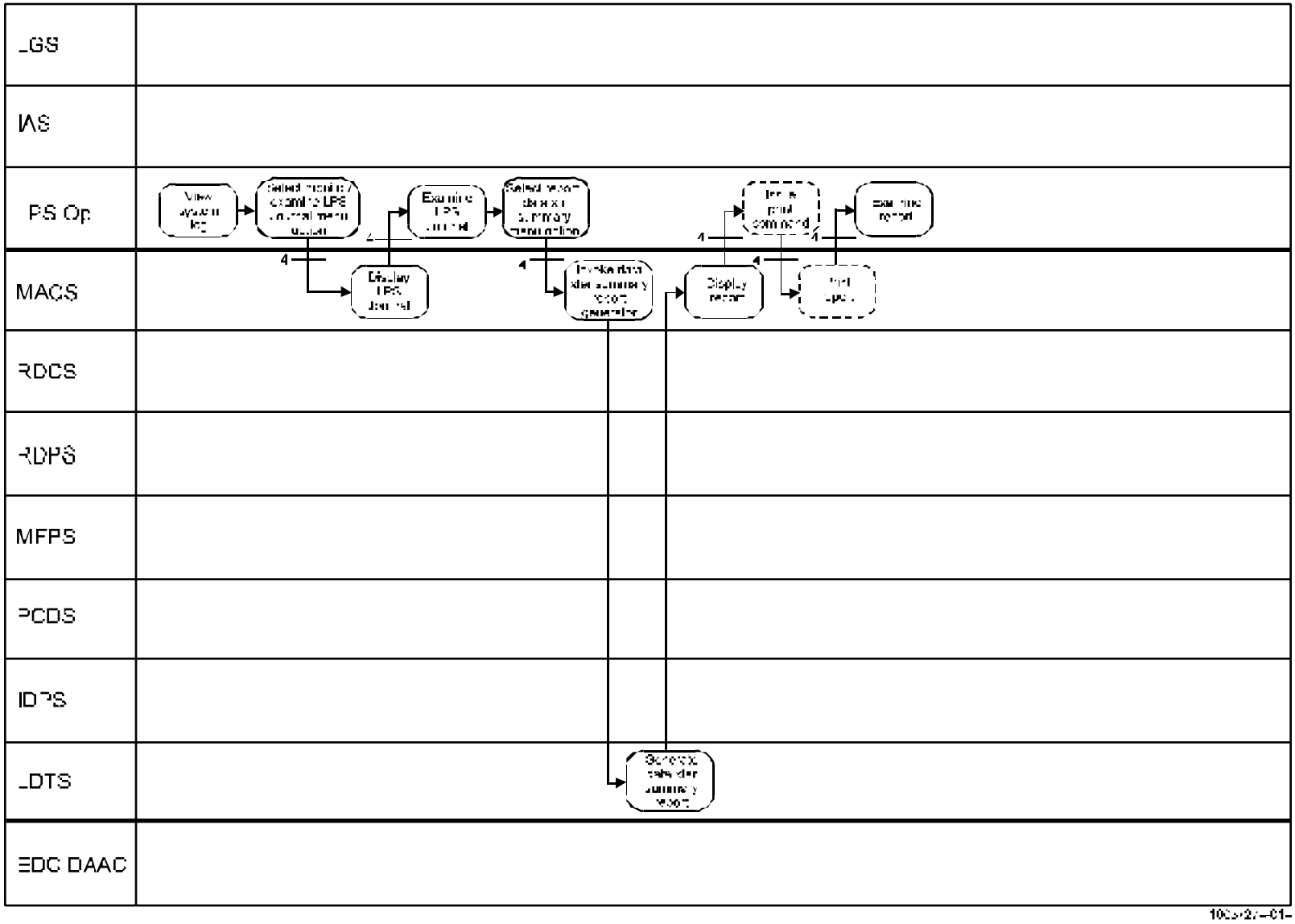


Figure 3–12. Monitor LPS-to-EDC DAAC File Transfers

2. LPS Operator—Select report/file transfer summary menu options, specifying time period of interest and display/print option.

3. MACS—Invoke file transfer summary report generator.

4. LDTS—Generate file transfer summary report for specified time period.

5. MACS—Display/print file transfer summary report.

6. LPS Operator—Examine report.

3.3.5 Restage Data for Reprocessing

The LPS operator initiates reprocessing for each contact period for which reprocessing has been requested by the IAS and that is still available in 30-day storage. Raw wideband data is restaged to available LPS strings. Thereafter, its processing is identical to newly captured raw wideband data. The steps performed to restage data for reprocessing (shown in Figure 3–13) are as follows:

1. IAS—Issue reprocessing request, specifying contact period to be reprocessed.

2. LPS Operator—Receive reprocessing request.

3. LPS Operator—Locate specified contact period in 30-day store.

4. LPS Operator—If contact period is not available, send notification to IAS.

5. LPS Operator—Schedule restage outside of contact schedule.

6. LPS Operator—Mount tapes.

7. LPS Operator—Select control/restage menu options, specifying tape drive and reprocessing version number.

8. LPS Operator—If necessary, restore calibration parameters for the contact period from disk.

9. MACS—Issue restage command to the RDCS.

10. RDCS—Copy contact information from tape into the LPS database.

11. RDCS—Copy contact period data file from tape to capture disk.

12. LPS Operator—After Level 0R processing is complete, restore any changed calibration parameters to the current values.

3.3.6 Support Operational Training and Test

Operational training and test support is provided by the backup LPS string and test console with DAN transmission capabilities disabled. Test data output by the LGS may be used to support training in data capture and Level 0R processing procedures. Backup copies of LPS database contents from active strings may be used to populate a training/test database. In addition, the backup string may be used to perform real-time raw data capture to populate its training/test database.

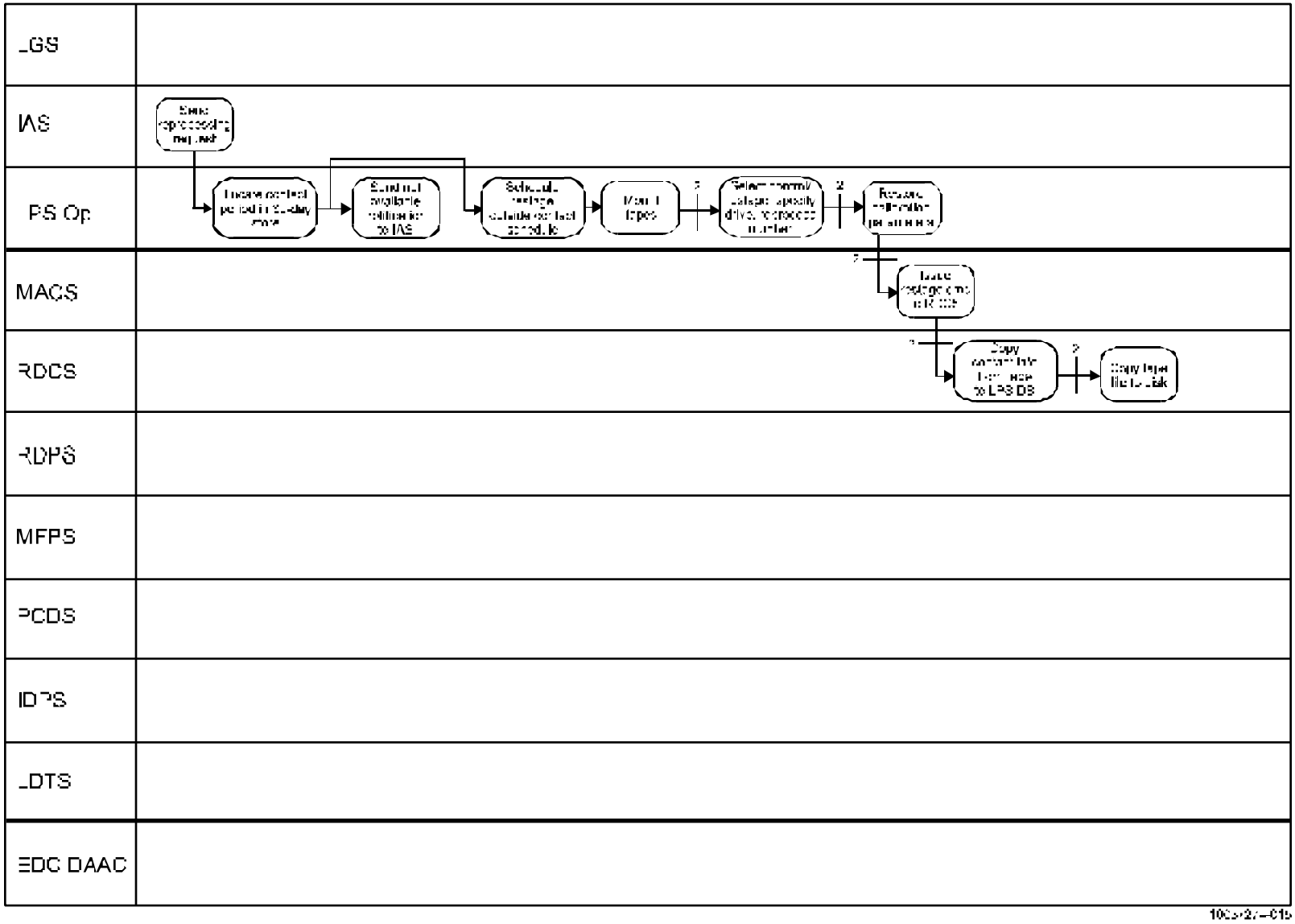


Figure 3–13. Restage Data for Reprocessing

The LPS operator can generate various reports that indicate data quality, file transfer status, etc. The steps performed to generate reports are as follows:

1. LPS Operator (optional)—Designate printer for reports.
2. LPS Operator—Select “Reports” from menu.
3. LPS Operator—Select appropriate report from menu items.
4. MACS—Generate and display report.

3.4 Contingency Operations

Contingency operations scenarios describe the sequences of operator activities performed to handle abnormal conditions during Landsat 7 data processing within the LPS. Abnormal conditions include software failures, hardware failures, and storage capacity shortfalls.

3.4.1 Receive Data From the LGS (Manual With Database)

The LPS operator may control data capture explicitly via commands to start/stop data capture using the MACS menu system. To execute, the MACS requires that the ORACLE DBMS be operational. The steps performed to capture raw wideband data manually (shown in Figure 3–14) are as follows:

1. LPS Operator—Select control/start capture menu option, specifying capture stop time.
2. MACS—Invoke RDCS with specified stop time.
3. RDCS—Begin data capture.
4. LPS Operator—Monitor data capture.
5. RDCS—Stop data capture.
6. RDCS—Insert data receive summary record into LPS database.
7. LPS Operator—Continue as in automatic capture (see Section 3.3.1, steps 7 through 15).

3.4.2 Receive Data From the LGS (Manual Without Database)

The LPS operator may initiate data capture directly from an IRIX shell command line. This may be required when the ORACLE DBMS has failed and the LPS user interface cannot be run. The steps performed to capture raw wideband data without database support (shown in Figure 3–15) are as follows:

1. LPS Operator—Review contact schedule for data capture time.
2. LPS Operator—Start data capture program, specifying data capture stop time.
3. RDCS—Initialize for data capture.
4. LGS—Begin sending data.
5. RDCS—Capture data.

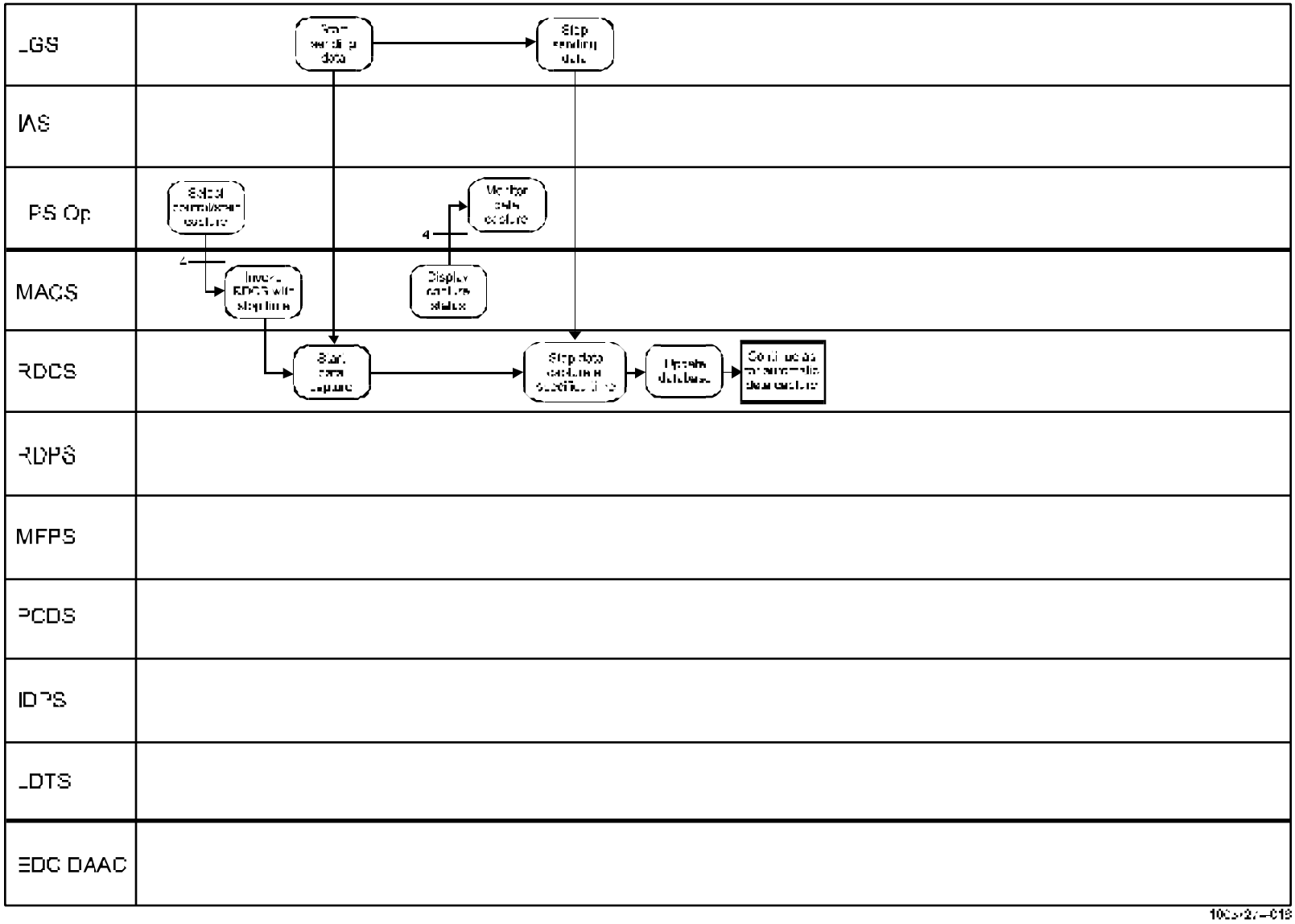


Figure 3–14. Receive Data From the LGS (Manual With Database)

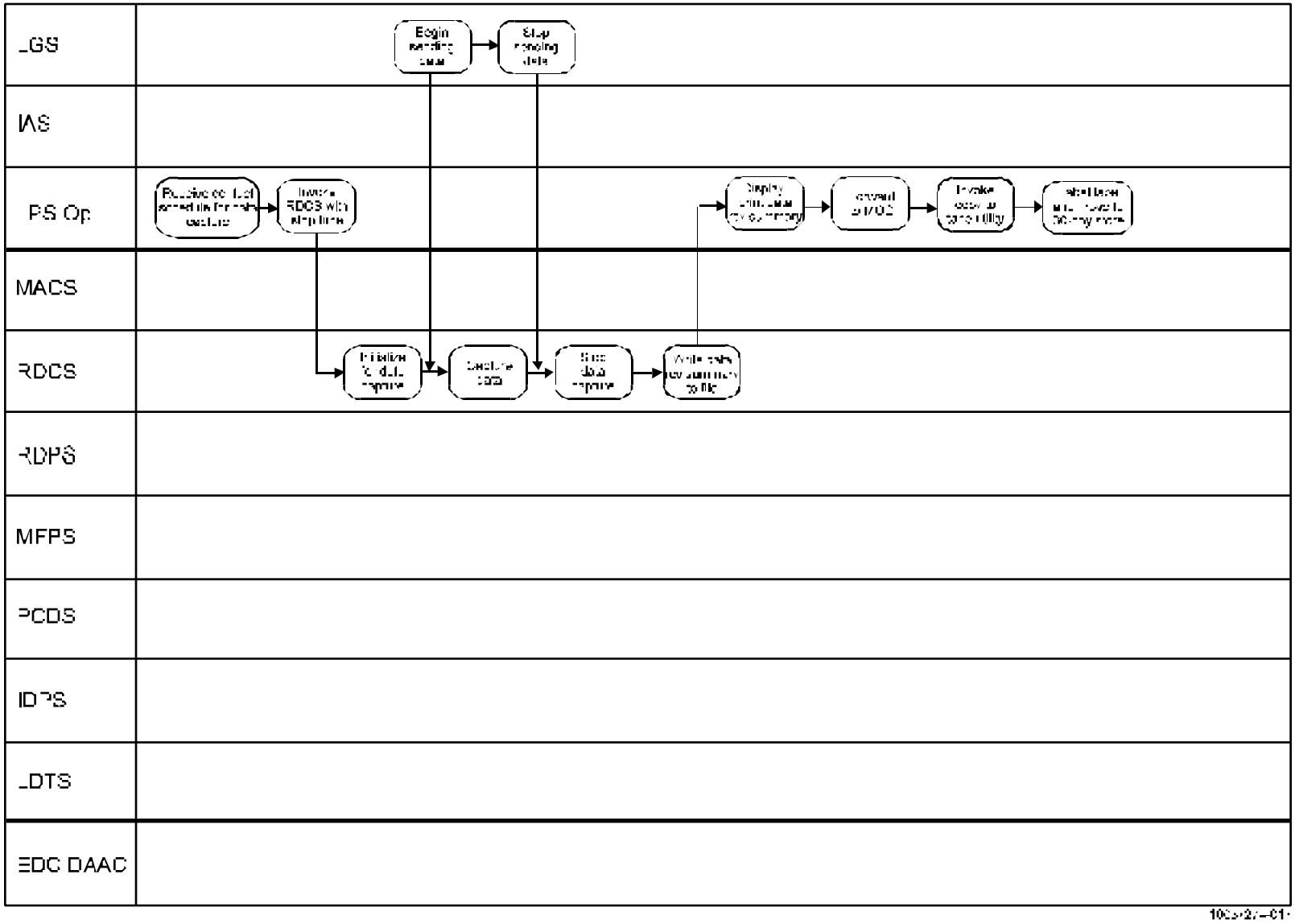


Figure 3–15. Receive Data From the LGS (Manual Without Database)

6. LGS Operator—Monitor data capture.
7. LGS—Stop sending data.
8. RDCS—Stop data capture at specified stop time.
9. RDCS—Write data receive summary information to file.
10. LPS Operator—Display/print data receive summary information file using IRIX utilities.
11. LPS Operator—Forward data receive summary to MOC.
12. LPS Operator—Mount tape and invoke copy to tape utility.
13. LPS Operator—Label tape and move to 30-day storage.

3.4.3 Restore Database After Non-Database Capture Operations

If the LPS operator has initiated data capture from the IRIX command line when the DBMS is not functioning, data receive summary information is not inserted into the LPS database. When the DBMS is functioning again, the LPS operator must move the data receive summary information from the files in which it has been stored by invoking the RDCS program for this purpose. The steps involved in restoring the LPS database after non-database capture operations (shown in Figure 3–16) are as follows:

1. LPS Operator—Invoke data receive summary loader program, specifying name of data receive summary file.
2. RDCS—Load specified data receive summary file into the LPS database.

3.4.4 Respond to Failure in LGS-LPS Interface

Either the LPS or LGS operator may detect an LGS-LPS interface failure. The steps performed by the LPS operator to respond to a failure in the LGS-LPS interface are as follows:

1. LPS Operator—If the LPS operator has detected the failure, notify the LGS operator; otherwise, receive notification from the LGS operator.
2. LPS Operator—If the failure occurs during data capture, notify the MOC of the data loss.
3. LPS and LGS operators—Resolve the problem in interface.

If the interface can be reestablished before the end of the contact period, two files will be present on the capture disk. The LPS operator may then be able to use a utility to concatenate the two files into one file with the appropriate start/stop times.

4. LPS Operator, MACS, RDPS, MFPS, PCDS, IDPS, LDTS (optional)—Continue Level 0R processing and transfer to EDC DAAC any data captured, including a partial or split contact period. (See Section 3.3.2 for the steps to perform Level 0R processing and Section 3.3.4 for the steps to transfer files to EDC DAAC.)

If the interface is down until after the contact period, use a test data set to verify the interface has been restored.

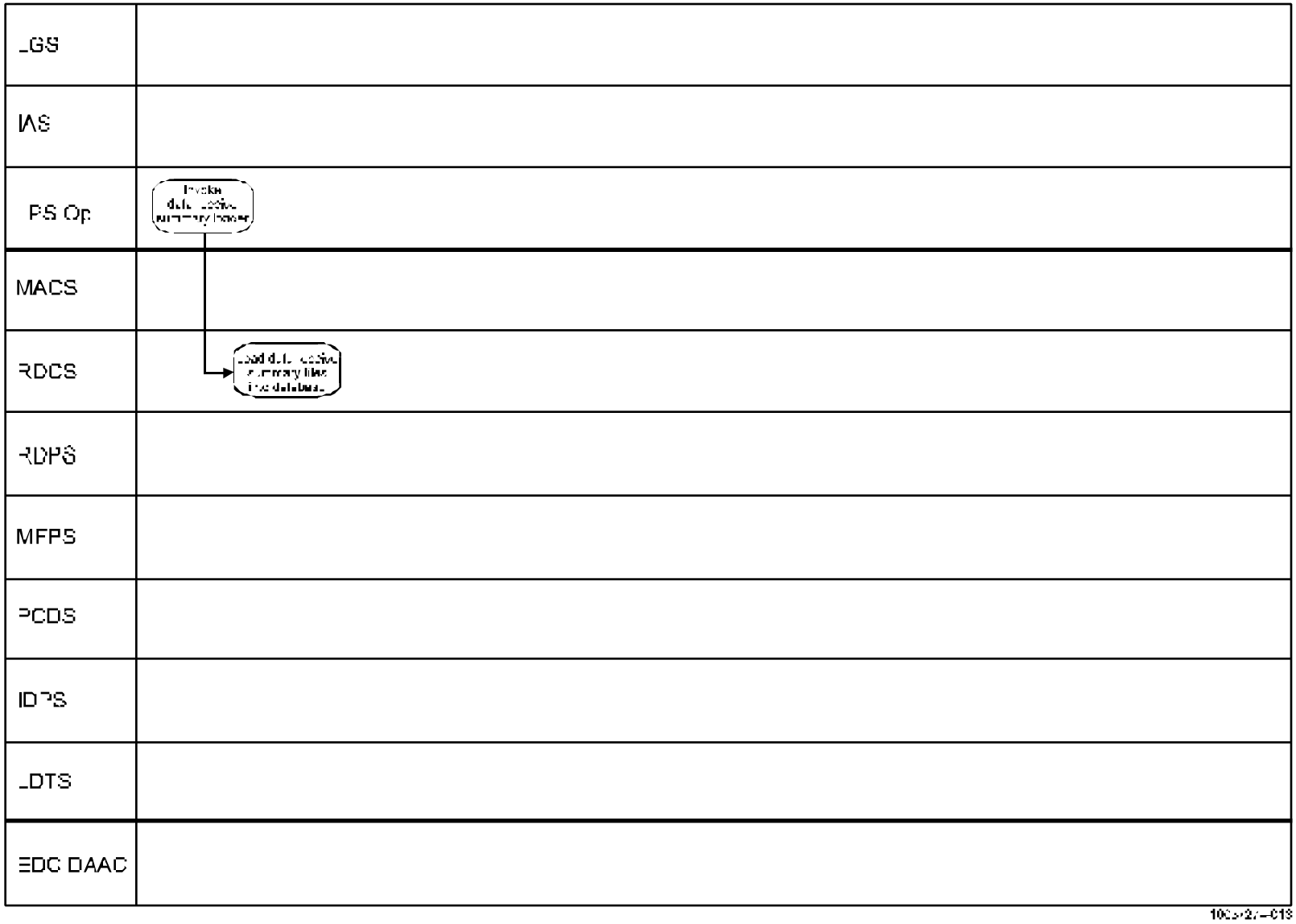


Figure 3–16. Restore Database After Non-Database Capture Operations

5. LPS Operator—Coordinate with the LGS operator to test the interface by capturing a test data set sent from the LGS. (See Sections 3.3.1, 3.4.1, and 3.4.2 for alternative data capture scenarios.)

6. LPS Operator—Verify the successful transmission of the test data set by direct examination of the file using IRIX utilities, test utilities, and/or through the LPS data receive summary. (See Section 3.3.1 for the steps to generate the data receive summary.)

7. LPS Operator—Delete transmitted test files and database entries for test data files.

3.4.5 Respond to Failure in LPS-EDC DAAC Interface

The LPS operator may detect an LPS-EDC DAAC interface failure while monitoring LPS disk space available for transfer storage. The LPS operator may also be notified of the failure by the EDC DAAC operator. The steps performed by the operator to handle this condition are as follows:

1. LPS Operator—If the LPS operator has detected the failure, notify the EDC DAAC operator; otherwise, receive notification from the EDC DAAC operator of the failure.

2. LPS Operator—Do not perform additional Level 0R processing.

3. LPS Operator—Continue to capture data and to copy the captured data to the 30-day store. Ensure sufficient disk capacity by deleting online capture data sets copied to the 30-day store. (See Sections 3.3.1, 3.4.1, and 3.4.2 for alternative data capture scenarios.)

4. EDC DAAC—Notify LPS operator that interface is restored.

5. LPS Operator—Resume Level 0R processing, restaging data from the 30-day store as necessary. (See Section 3.3.6 for the steps to restage data.) The outstanding DANs at the time the interface failed should not have to be resent unless DAAs were not received from EDC DAAC. Outstanding DDNs from EDC DAAC would only have to be resent if DDAs were not received from LPS.

3.4.6 Respond to Exhaustion of LPS Output Storage Capacity

The LPS operator detects that LPS output (back end) storage capacity has been exceeded while monitoring the output storage available on LPS strings. The steps performed by the operator to handle this condition are as follows:

1. LPS Operator—Continue receiving data from the LGS as scheduled.

2. LPS Operator—Do not initiate Level 0R processing on any received data until output storage is available. Coordinate with the EDC DAAC operator in regard to the EDC DAAC schedule for transferring retained files.

3.4.7 Respond to LPS String Failure

LPS string failure occurs whenever a LPS string data process hardware configuration item (HWCI) or any of its peripheral HWCIs fails. The failure of any part of the string is treated identically to a failure of the whole. The steps performed to handle LPS string failures (shown in Figure 3–17) are as follows:

1. LPS Operator—Coordinate with the LGS operator to switch the LGS output channel received by the failed string to the backup string.
2. LGS Operator—Switch the channel to the backup string.
3. LPS Operator—Notify the EDC DAAC operator of the failure.
4. LPS Operator (optional)—If necessary, start the LPS backup string. (See Section 3.2.1 for the steps to start a string.)
5. LPS Operator—Ensure the string is configured with the current software version.
6. LPS Operator—Set up the string for data capture. (See Section 3.2.3 for the steps to set up for data capture.)
7. LPS Operator—Select setup/modify schedule menu options and enter contact schedule information for a new LGS channel. (See Section 3.2.2 for the steps to modify the contact schedule.)
8. LPS Operator, MACS, and RDCS—If the string failed after raw data capture and before completion of other processing, mount archived, unprocessed raw data tapes from the failed string and copy to backup. (See Section 3.3.6 for the steps to restage data from tape.)
9. LPS Operator, MACS, RDPS, MFPS, PCDS, IDPS, and LDTS—Process restaged and newly captured raw data to Level 0R. (See Section 3.3.2 for the steps to perform Level 0R processing.)

3.4.8 Restore LPS String

Once a failed string has been restored, the following steps (shown in Figure 3–18) are used to make it operational:

1. LPS Operator—On the backup string, select the reports/data transfer summary menu options to invoke data transfer summary report generation.
2. MACS—Invoke the data transfer summary report generator.
3. LDTS— Generate the data transfer summary report.
4. MACS—Display/print the data transfer summary report.
5. LPS Operator—If all files on backup string have not been transferred, resend any outstanding DANs.
6. LPS Operator—Coordinate with the LGS operator to switch the LGS channel from the backup string to the restored string.
7. LGS—Switch the LGS channel to the restored string.
8. LPS Operator—Start up the restored string. (See Section 3.2.1 for the steps to start up an LPS string.)

Figure 3–17. Respond to LPS String Failure

_GS
IAS
PS Op
MACS
RDCS
RDPS
MFPS
PCDS
IDTS
_DTS
EDC DAAC

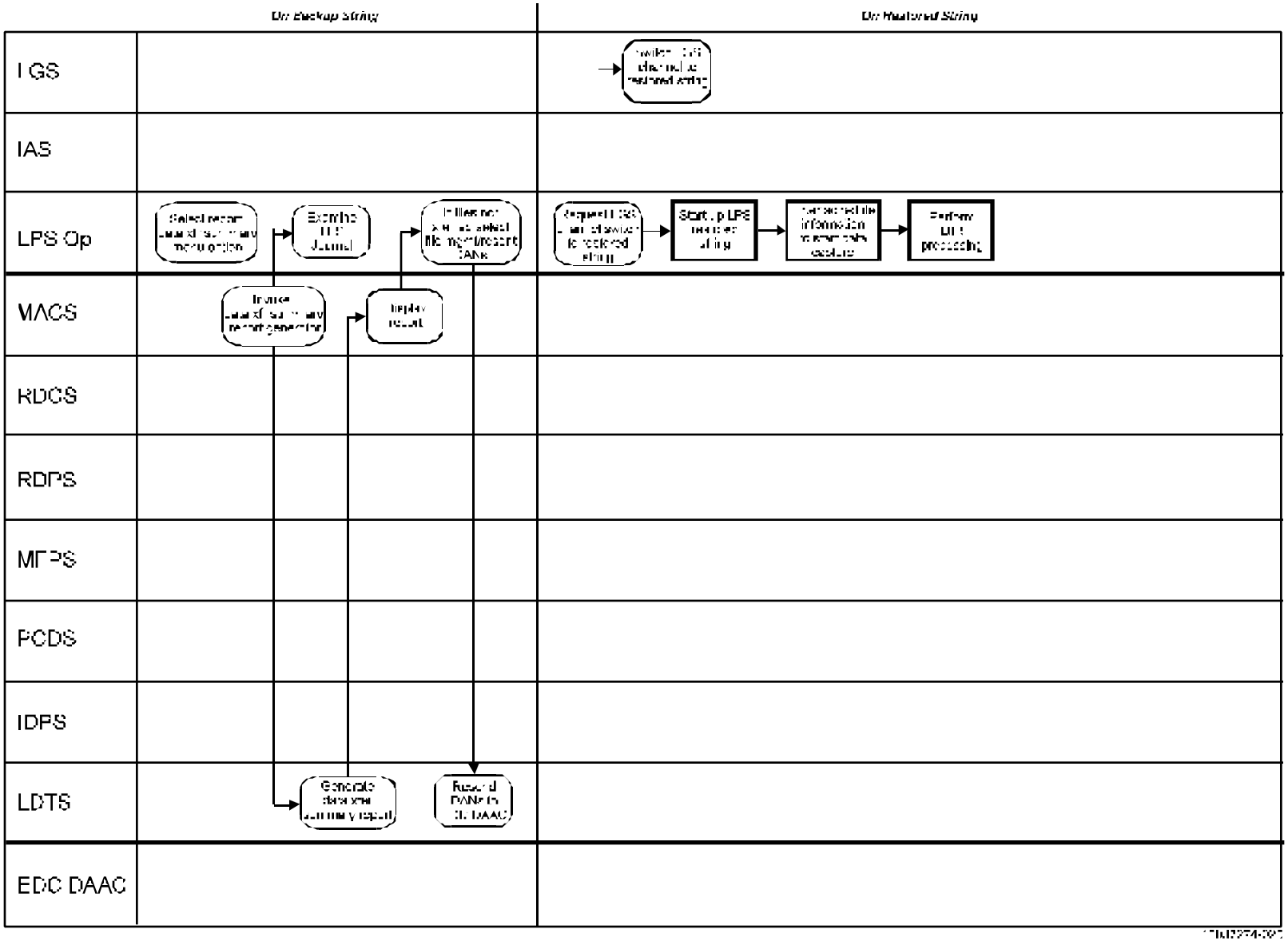


Figure 3–18. Restore LPS String

9. LPS Operator—Enter the contact schedule information on the restored string. (See Section 3.2.2 for the steps to enter contact schedule information.) Continue automatic data capture on the restored string. (See Section 3.2.1 for the steps to perform automatic data capture.)

10. LPS Operator—Perform Level 0R processing on the restored string. (See Section 3.3.2 for the steps to perform Level 0R processing.)

Section 4. Global Libraries

4.1 Introduction

The LPS will use global libraries to implement the interfaces between subsystems. These libraries contain the global routines, global files, and global data. They not only reduce the duplicate implementation among the subsystems, but also hide the system configuration from the application software for easier future software porting purposes. The global libraries also provide the simplicity of design within the subsystems when the subsystems use the global functions.

4.2 Design Overview

This section provides an overview of the global library design. It is presented along with a discussion of the assumptions and considerations used in the design process.

4.2.1 Library Overview

As mentioned previously, the global libraries are those candidates in which either the functions are shared by Landsat 7 subsystems or are system hardware dependent and need to be isolated for future software porting purposes, or that can provide the simplicity of design to the interfacing subsystems.

4.2.2 Design Considerations

This subsection presents the design drivers relevant to the global library software design. The major assumptions or considerations that influence the design of the global library software are as follows:

- The LPS interprocess communication (IPC) mechanisms will only use UNIX System V IPC for the consideration of portability.
- All global library routines are designed so that the application programming interfaces (APIs) will be kept to the minimum. On the other hand, the APIs from the global libraries to the operating system will be transparent when the global functions are implemented.
- Each global function will have its own structure chart and is invoked by the subsystems.
- The naming convention follows the LPS software naming convention prefixed with “lps” as a global identification.
- Global libraries are responsible for allocating and deallocating all FIFO queues and shared memories used in the LPS at the beginning and end of Level 0R processing. This will prevent any aborted subsystem or process from leaving the unwanted FIFO queues or shared memories in the LPS that eventually leads to the system resource starvation.

4.3 Library Design

The following sections list the LPS global routines that are shared by LPS subsystems.

4.3.1 LPS IPC Mechanisms

The LPS subsystems will use UNIX System V IPC mechanisms to communicate during LPS data processing. Three mechanisms are used in LPS processing: message queue, semaphore, and shared memory. The message queue is developed under LPS FIFO queue operation routines. Shared memory is developed under LPS shared memory resource management routines. The semaphore is developed on top of the shared memory to control the flow of shared memory.

Because the global libraries are responsible for allocating the FIFO queues and the shared memories used in the LPS, MACS first (when MACS is brought up) invokes lps_RsrcAlloc() to start creating interprocess communication resources. It creates an LPS shared-memory resource control shared-memory segment and an LPS FIFO control shared memory segment. These two shared-memory segments are used to manage the LPS shared-memory resources and the LPS FIFO resources. They also contain the resource access information for LPS subsystems. Once the control shared-memory segments have been created, the lps_RsrcAlloc() invokes lps_RsrcAllocFIFO() to create LPS FIFO queues and invokes lps_RsrcAllocShm() to create LPS shared-memory resources.

When Level 0R data processing is complete or when an error is encountered during Level 0R data processing, MACS invokes lps_RsrcDealloc(), which internally removes the LPS FIFO queues, and also invokes lps_ShmRemove() to remove the LPS shared memories from the system. This guarantees that no unwanted FIFO queues or shared memories will be left in the system.

All LPS IPC resources will be assigned unique keys that are constant and are defined in the global libraries. MACS uses the keys to get the effective keys, which are then used to create/remove the LPS IPC resources. The other subsystems or processes provide the assigned keys to the global libraries, and the global libraries map out the effective keys for the subsystems to gain access to the LPS IPC resources. Figure 4–1 contains the LPS Resource Structure Chart.

4.3.1.1LPS Shared Memory Resource Management

Once the shared resource control shared-memory segment is created, the lps_RsrcAllocShm() invokes the lps_ShmCreate() to create the shared-memory segments used in the LPS. Based on the specific number of shared memories used by the subsystems, the number of blocks in each shared memory, and the block size needed by the subsystems, lps_ShmCreate() creates the shared-memory segments. After the shared-memory segments are successfully created, a semaphore set is created accordingly for each shared-memory segment and is used to control and synchronize the access of shared memory blocks. The created shared-memory segments are then divided into several fixed-size blocks and the block addresses, and associated shared-memory segment and information are stored in the shared resource control information table. Initially, all the shared-memory blocks are initialized as free (write) blocks in the block pool. The subsystem calls lps_ShmGetWrBlk() to retrieve a free block for writing and calls lps_ShmPutWrBlk() to put the written block into the active (read) block pool after it finishes the write block.

Likewise, when a subsystem needs to retrieve information from the active (read) block pool, it calls lps_ShmGetRdBlk() to get a read block and calls lps_ShmPutRdBlk() to return the read block back into the free block pool. The lps_ShmAddListTail(), lps_ShmRemListTail(), and

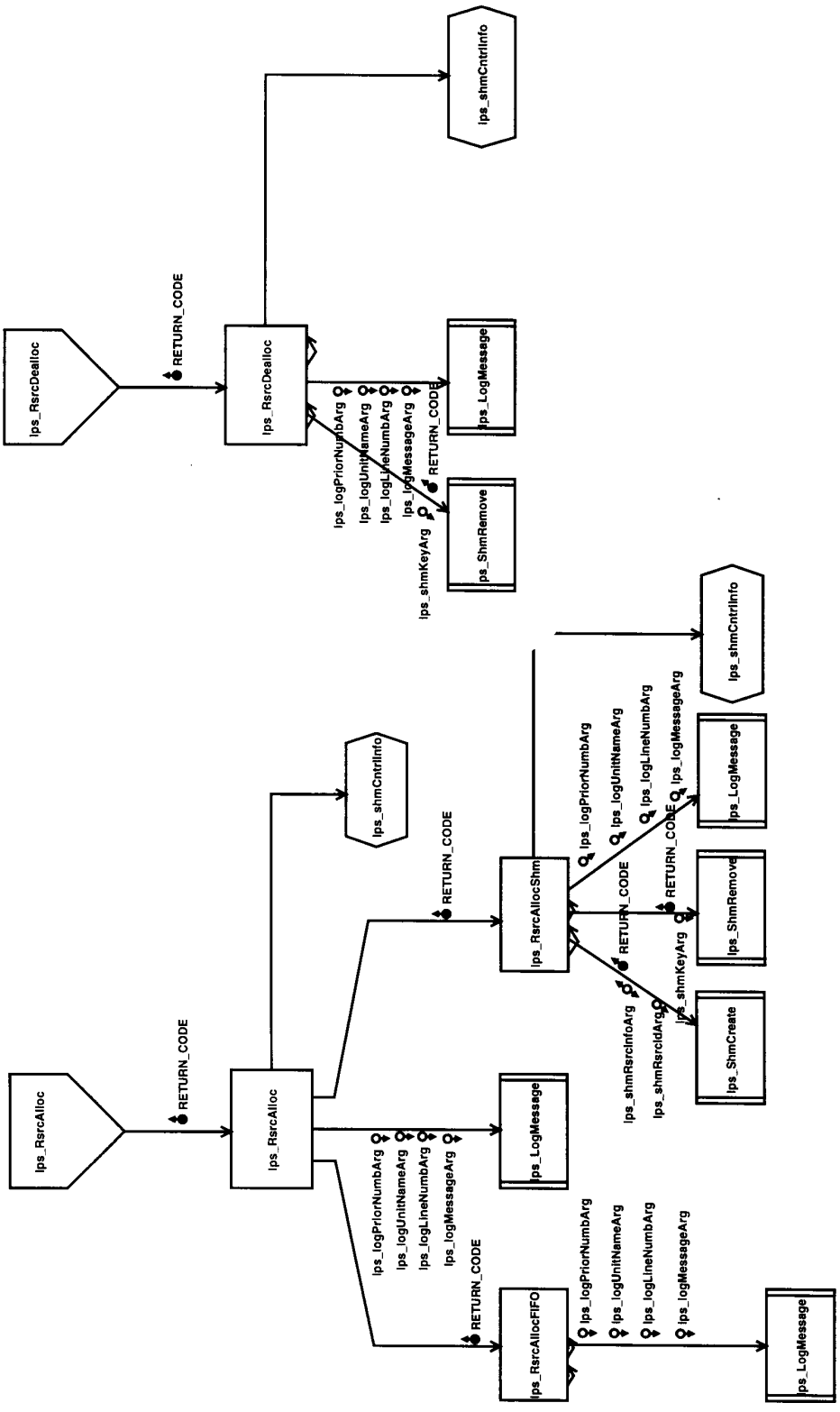


Figure 4–1. LPS Resource Structure Chart

lps_ShmRemListHead low-level global library functions are used to insert and remove the shared-memory blocks in the pools. The blocks in the free and active pools are managed in a first-in-last-out (FILO) pattern. On top of the FILO control mechanism, the blocks are guarded by the semaphores to prevent the racing condition among the subsystems. The semaphores also are used to give the subsystems the option of whether or not to wait for a block to become available. Figure 4–2 contains the shared-memory structure charts.

4.3.1.2LPS FIFO Queue

The LPS FIFO queue contains four routines (Figure 4–3). The subsystems or processes requiring access to the LPS FIFO queues will first call lps_FIFOOpen() to attach to the LPS FIFO queues created by the MACS. The subsystems or processes can then call lps_FIFOSend() and lps_FIFOReceive() to exchange information. In the meantime, the subsystems or processes can specify whether or not to wait when reading or writing from/to the LPS FIFO queues. After data processing is done or an error is encountered, the subsystems or processes call lps_FIFOClose() to detach themselves from LPS FIFO queues.

4.3.2 LPS File-Related Operation

The LPS file-related operation functions are used by the subsystems to obtain various attributes pertaining to the LPS data files:

- lps_GetPIDFileName() provides the name of the temporary file that was created by the capture process of the RDCS. The existence of such a file signifies RDCS activity.
- lps_FileNameCreate() provides the ability to create the LPS Level 0R output filenames, including a full path from the specified file type and subinterval identifier.
- lps_ParseFileName() provides three attributes of a Level 0R filename: the file path, filename, and file type.
- lps_ValidateRDCOutfileName() parses and validates the raw data capture filename.

Figure 4–4 contains structure charts of LPS file-related operation functions.

4.3.3 LPS Process Status, Initialization, and Handling

The LPS process status, initialization, and handling routines are created to facilitate the management and spawning of child processes. They include LPS process initialization, lps_ProcessInit(); child process creation, lps_ProcessStartChild(); child exit status reporting, lps_ProcessChildStatus(); child argument process and handling, lps_ParseOptions and lps_GetOpt; and indication as if the RDCS capturing process is running, lps_CaptureIsRunning(). The structure charts are shown in Figure 4–5.

4.3.4 LPS Message Logging

The LPS will be using basic UNIX system services to provide status and error reporting. Each subsystem will be calling the lps_LogMessage() routine that “sends” the status or error to the MACS via the UNIX syslog function. This function also provides the option to log the messages

to the standard error file in case the UNIX syslog function is failed. Figure 4–6 contains the structure chart of LPS Message Logging.

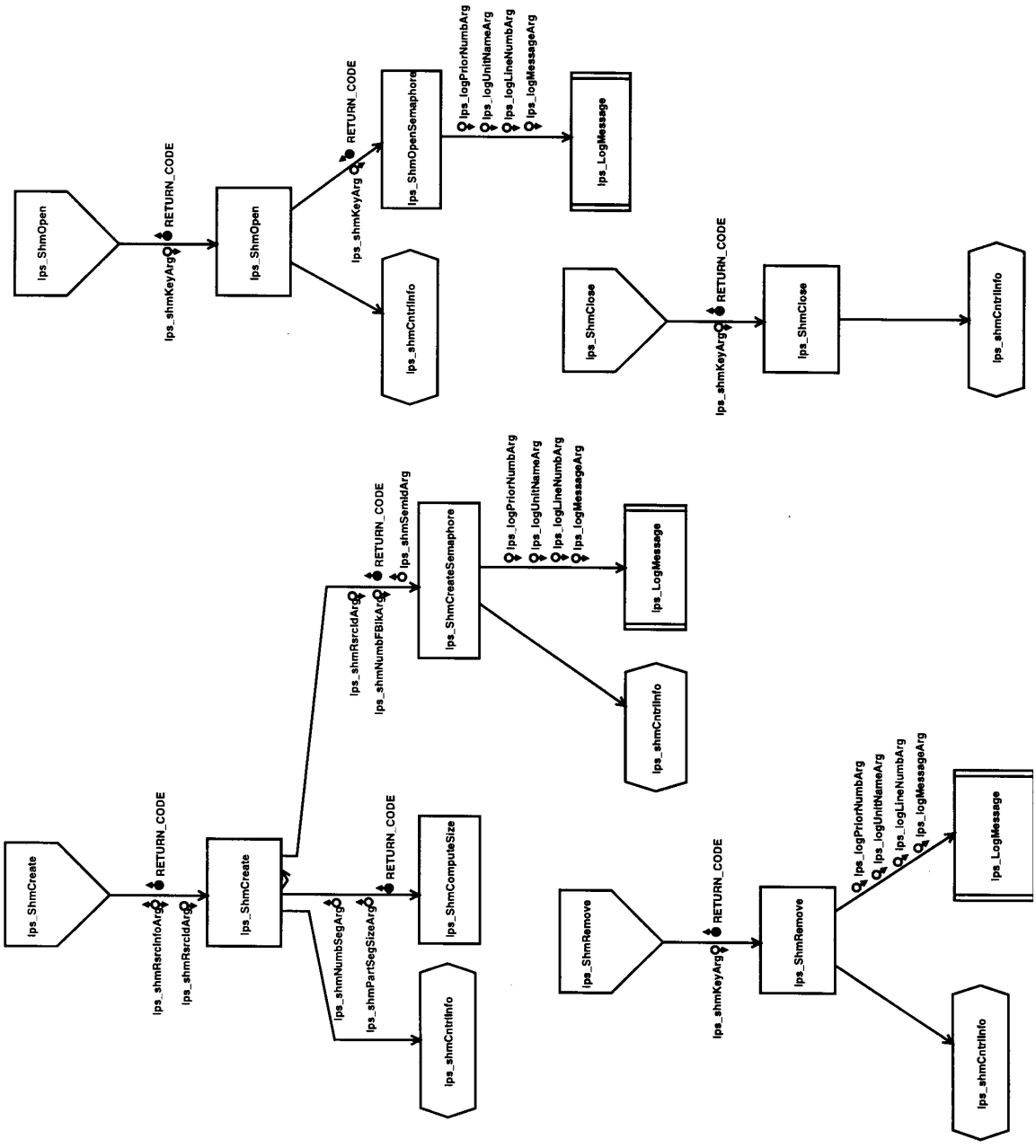


Figure 4–2. Shared-Memory Structure Chart (1 of 3)

Figure 4–2. Shared-Memory Structure Chart (2 of 3)

Figure 4–2. Shared-Memory Structure Chart (1 of 3)

Figure 4–3. FIFO Structure Chart

Figure 4–4. File-Related Operation Structure Chart

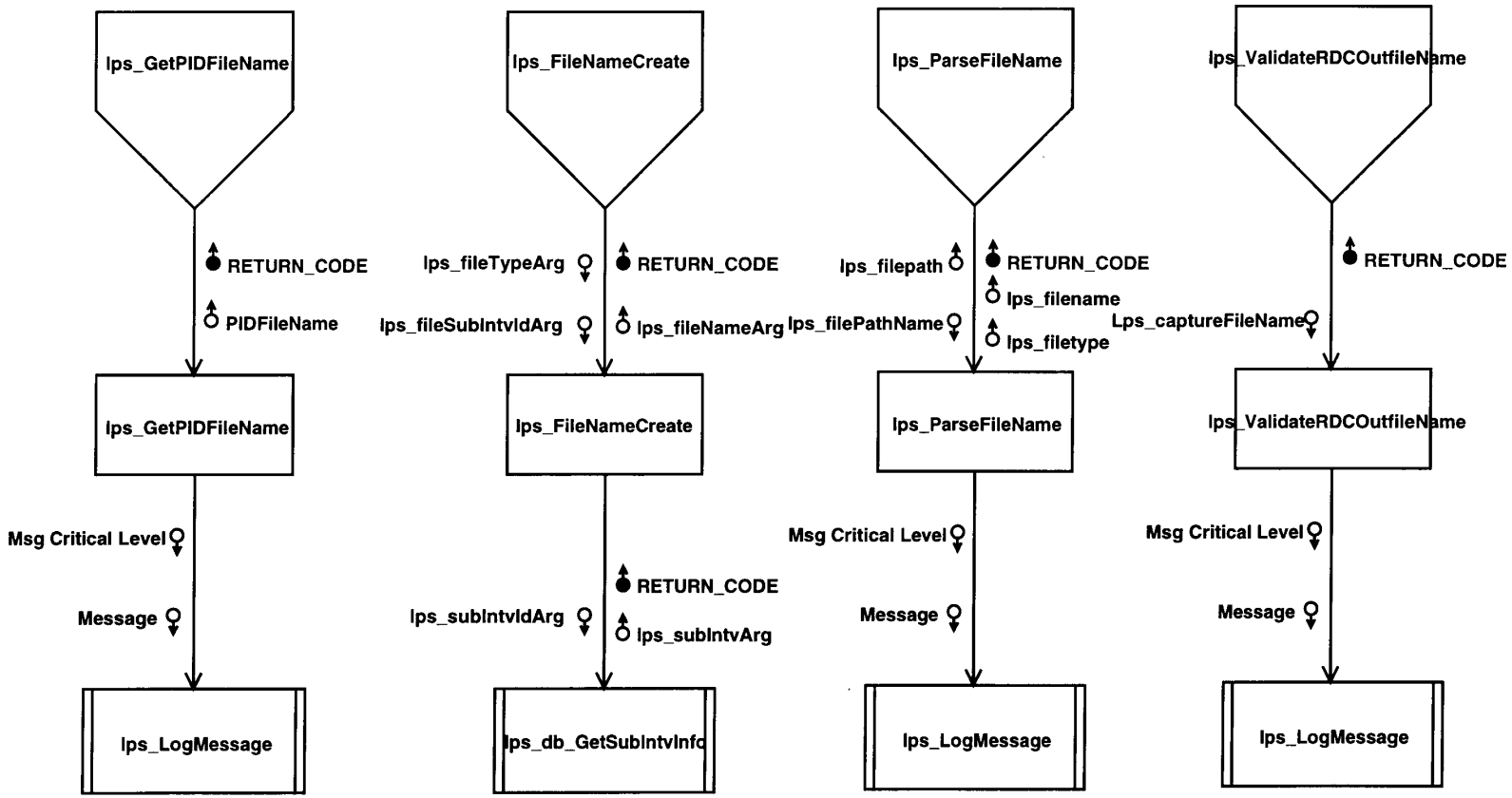


Figure 4–5. LPS Process Status, Initialization, and Handling Structure Chart

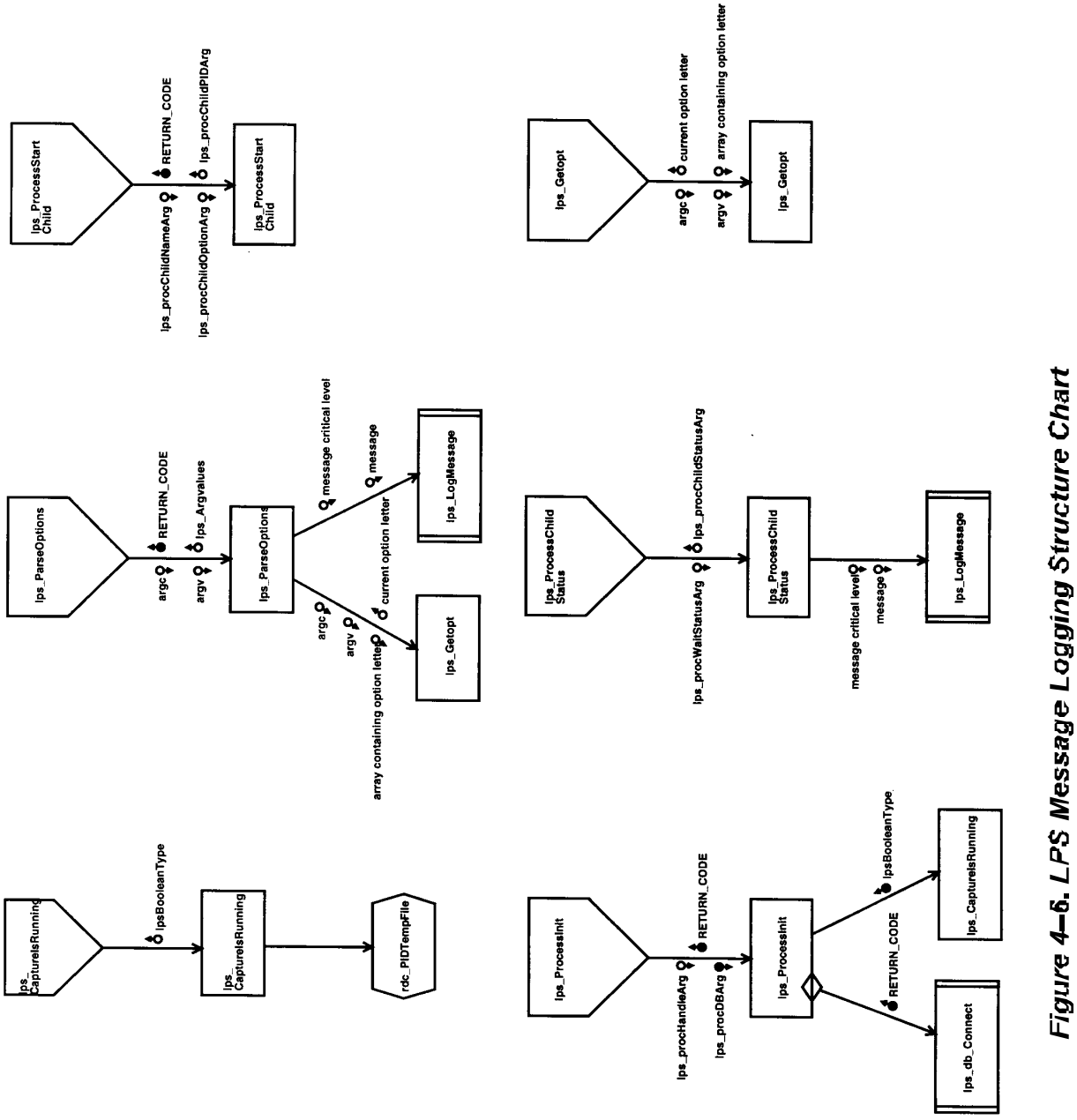
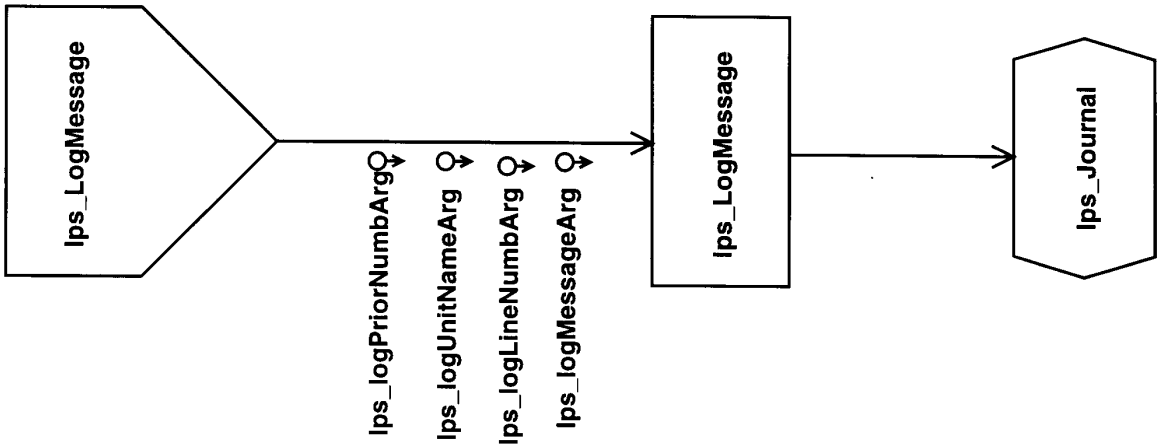


Figure 4–6. LPS Message Logging Structure Chart


lps_LogMessage
lps_logPriorNumbArg
lps_logUnitNameArg
lps_logLineNumbArg
lps_logMessageArg
lps_LogMessage
lps_Journal

4.3.5 LPS Time Manipulation

The LPS subsystems will be using various global time routines to provide the functionality in the LPS Level 0R data processing. These routines include obtaining and retrieving system time in specific format, comparing two different times, adding two different times, dividing two time durations, and converting various time formats. Figure 4–7 contains the structure charts for the LPS global time routines.

4.3.6 LPS Database Access

The LPS global database access routines provide the common functions for accessing the LPS database, accessing database tables shared by more than one subsystems, and registering Level 0R output files in the database. The LPS database access functions structure charts are shown in Figure 4–8.

4.3.7 LPS Data Capture Status

The LPS Data Capture Status routine (shown in Figure 4–5), namely lps_CaptureIsRunning, allows the LPS subsystems to verify if the LPS Data Capture function is active. Most LPS functions should suspend themselves when the Data Capture function is running.

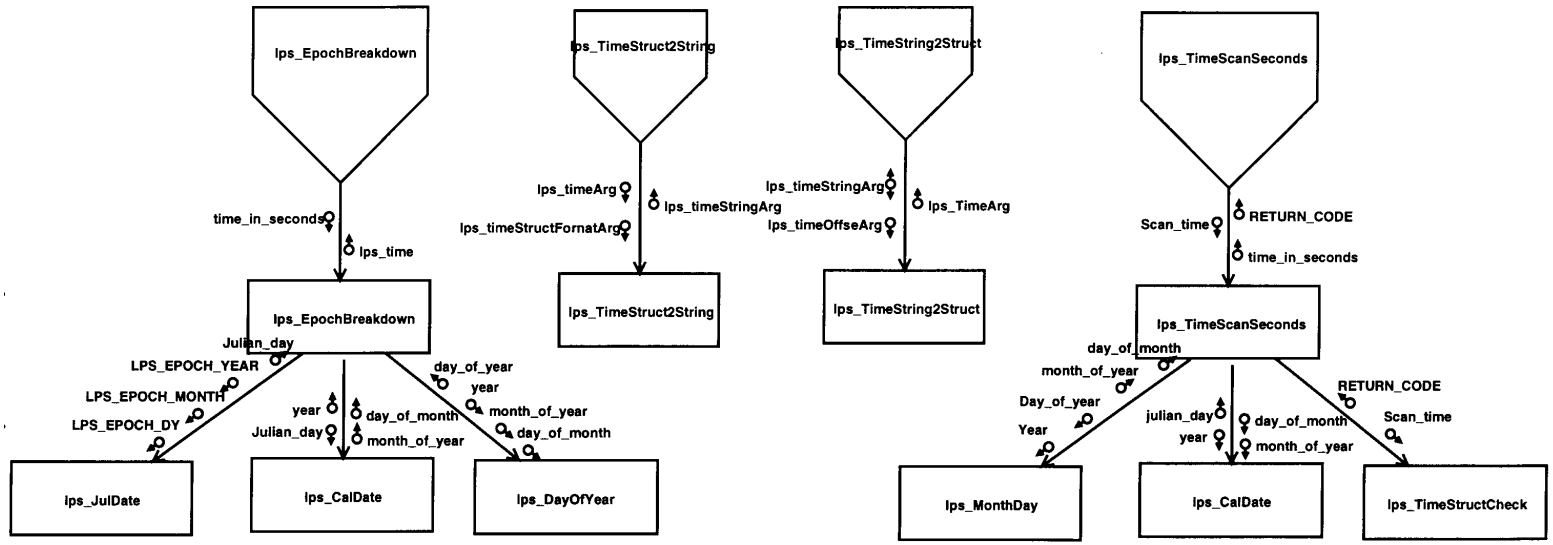


Figure 4–7. LPS Time Manipulation Structure Chart (1 of 3)

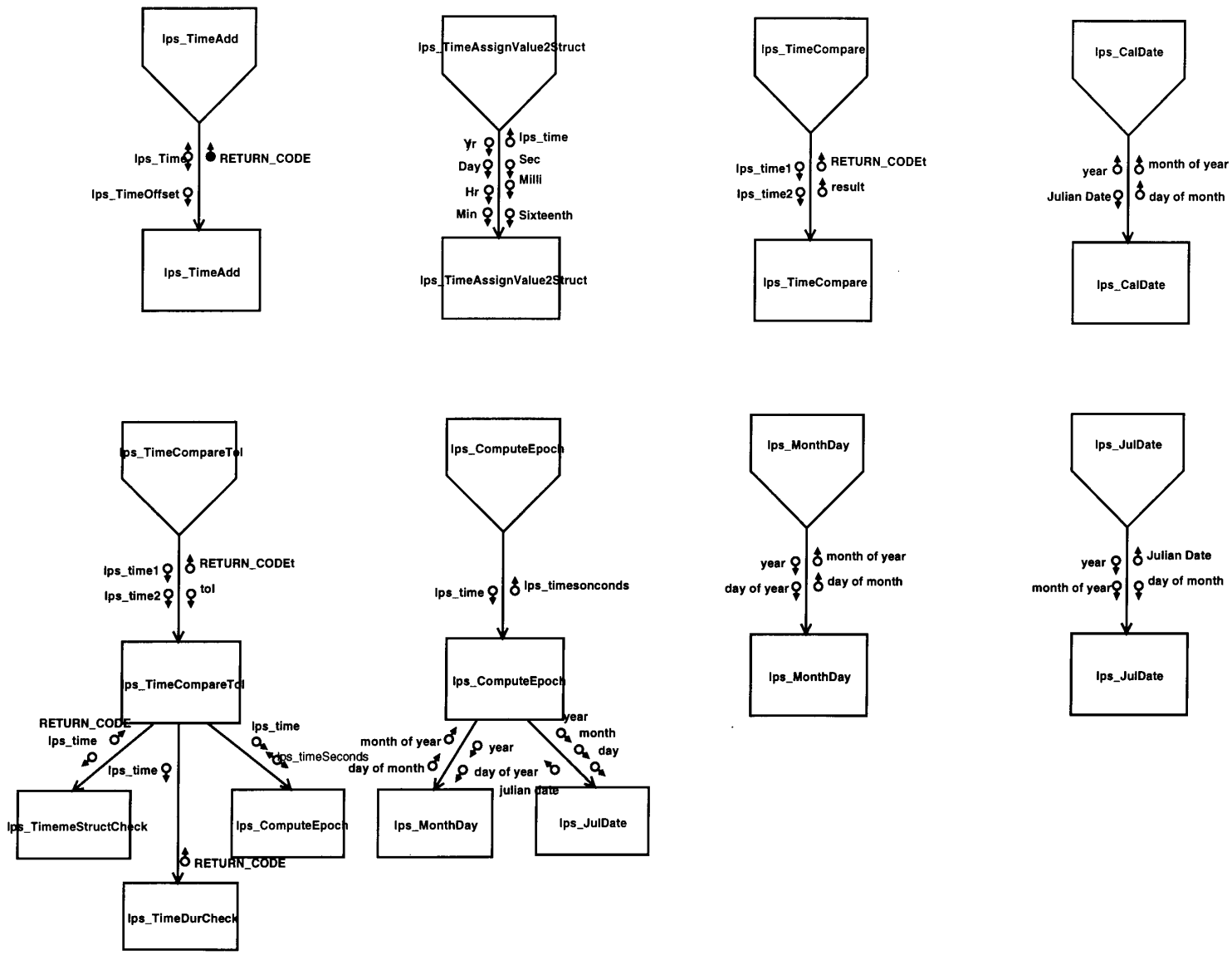


Figure 4–7. LPS Time Manipulation Structure Chart (2 of 3)

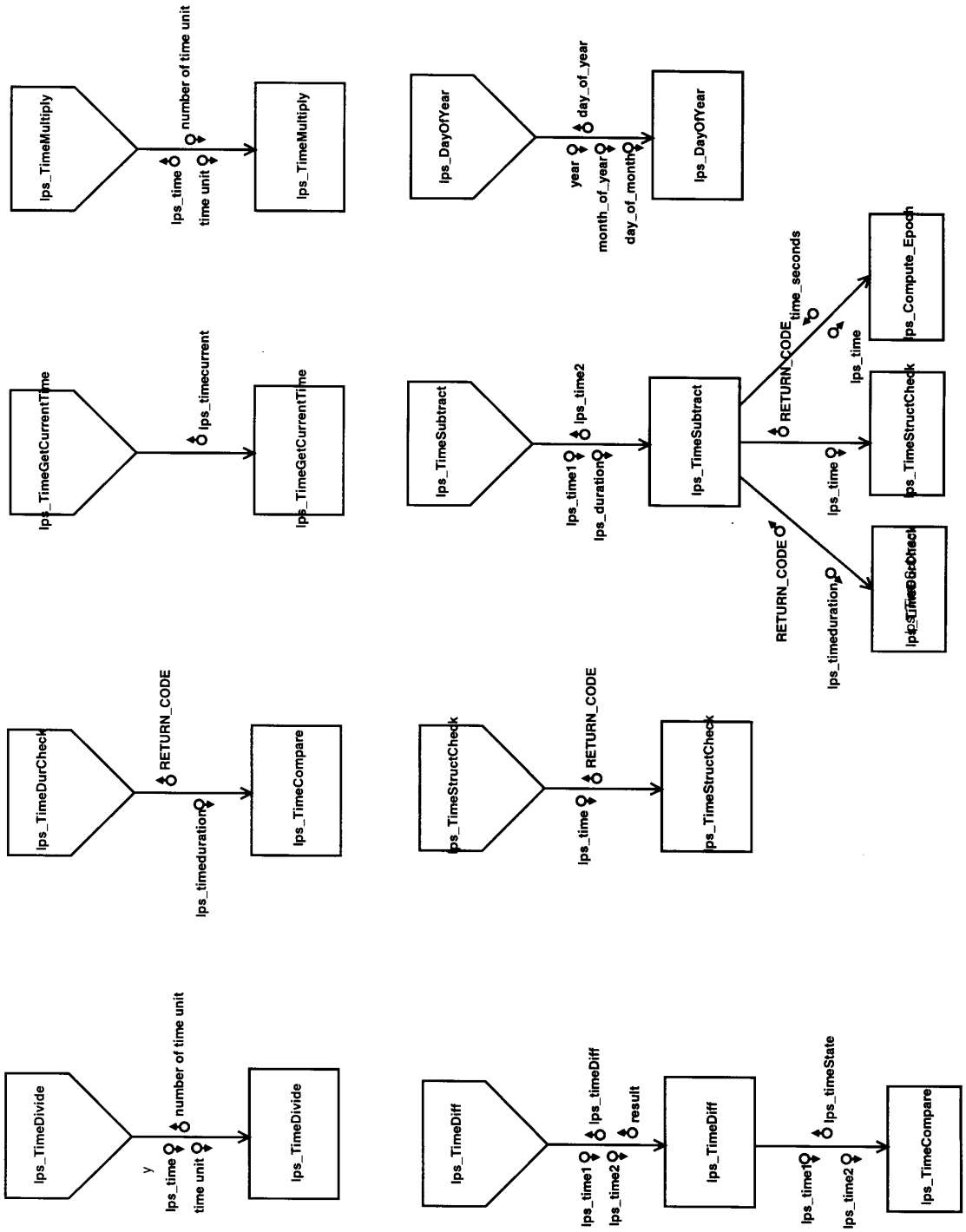


Figure 4–7. LPS Time Manipulation Structure Chart (3 of 3)

Figure 4–8. LPS Database Access Structure Chart (1 of 3)

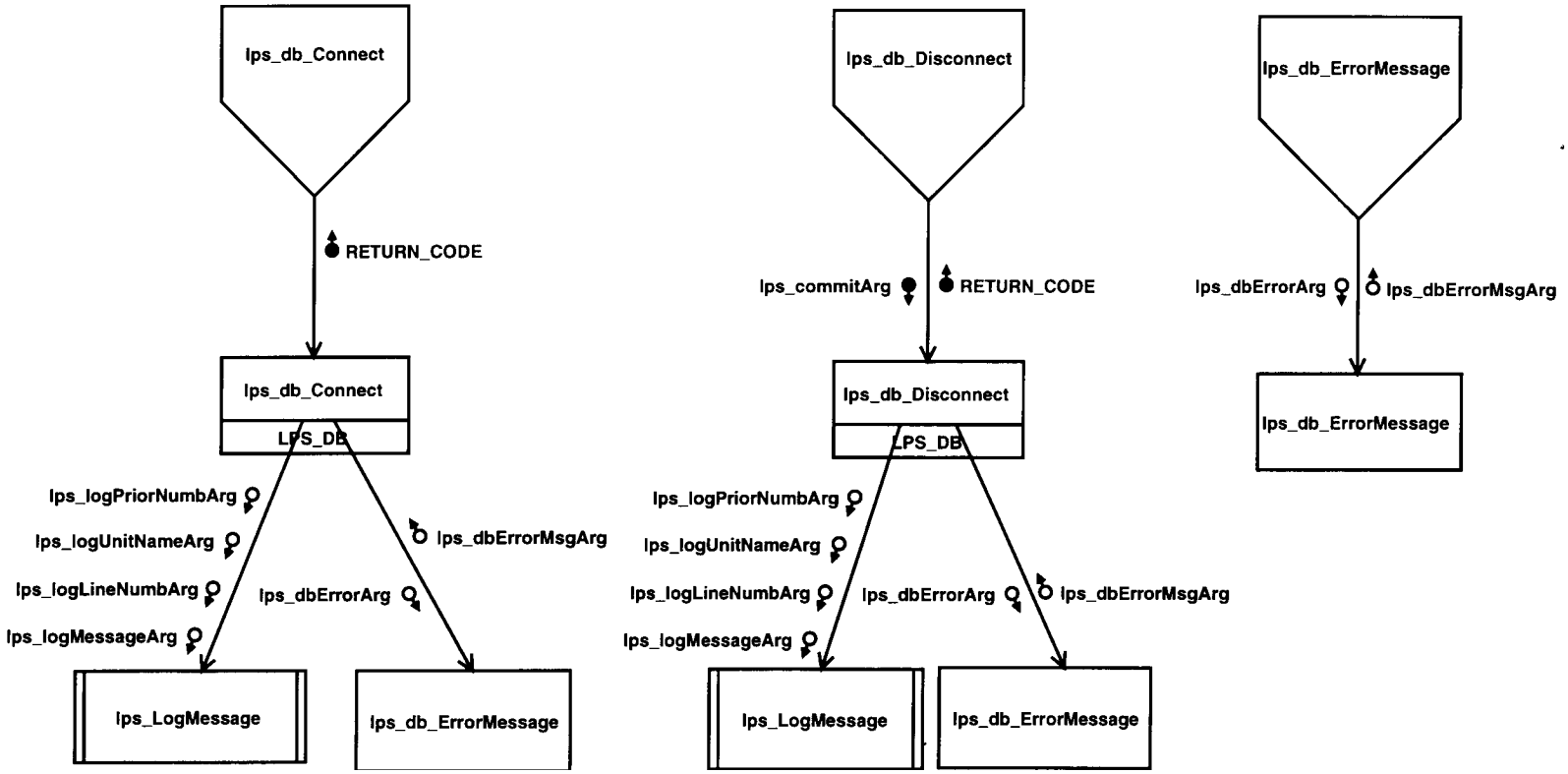


Figure 4–8. LPS Database Access Structure Chart (2 of 3)

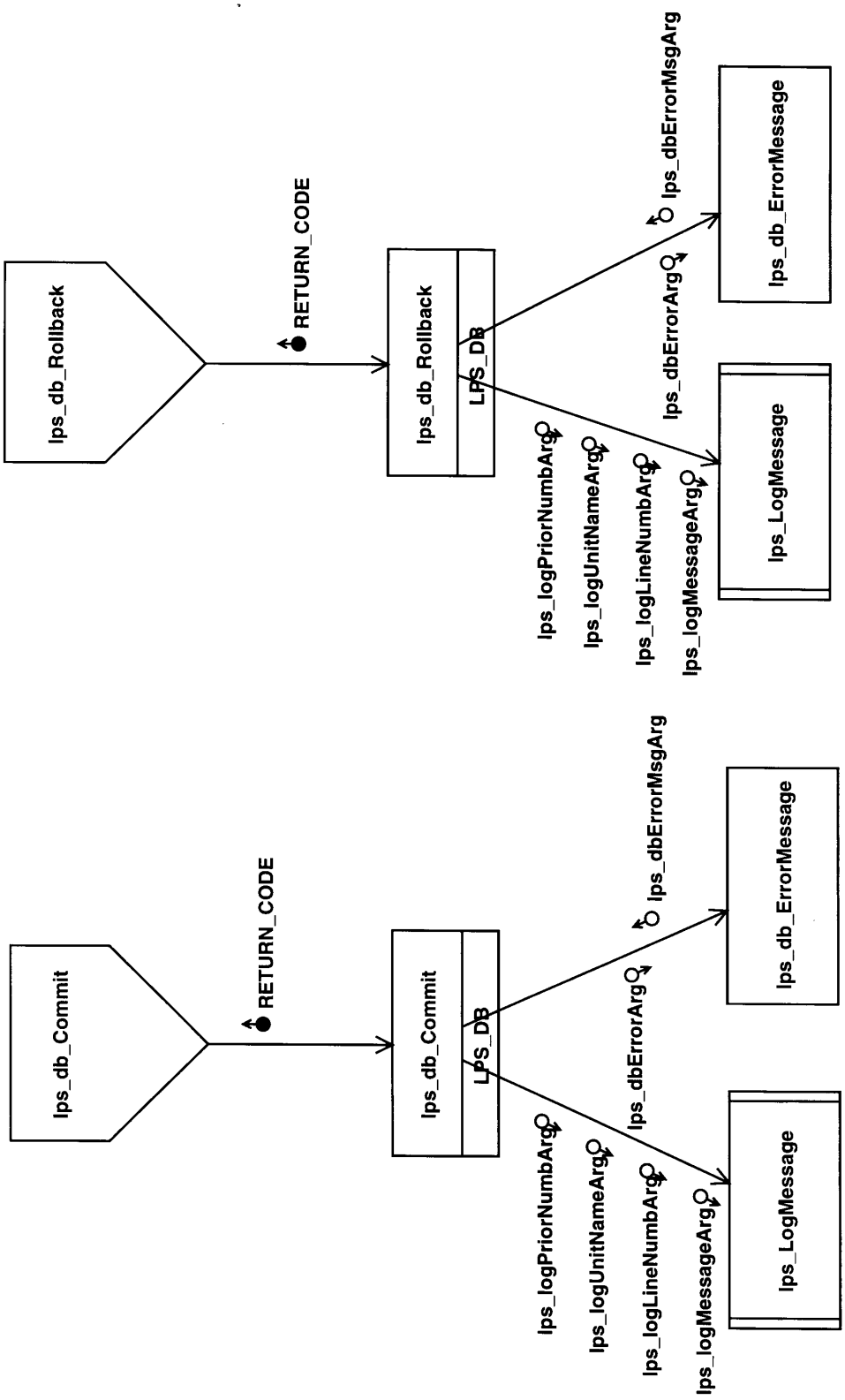
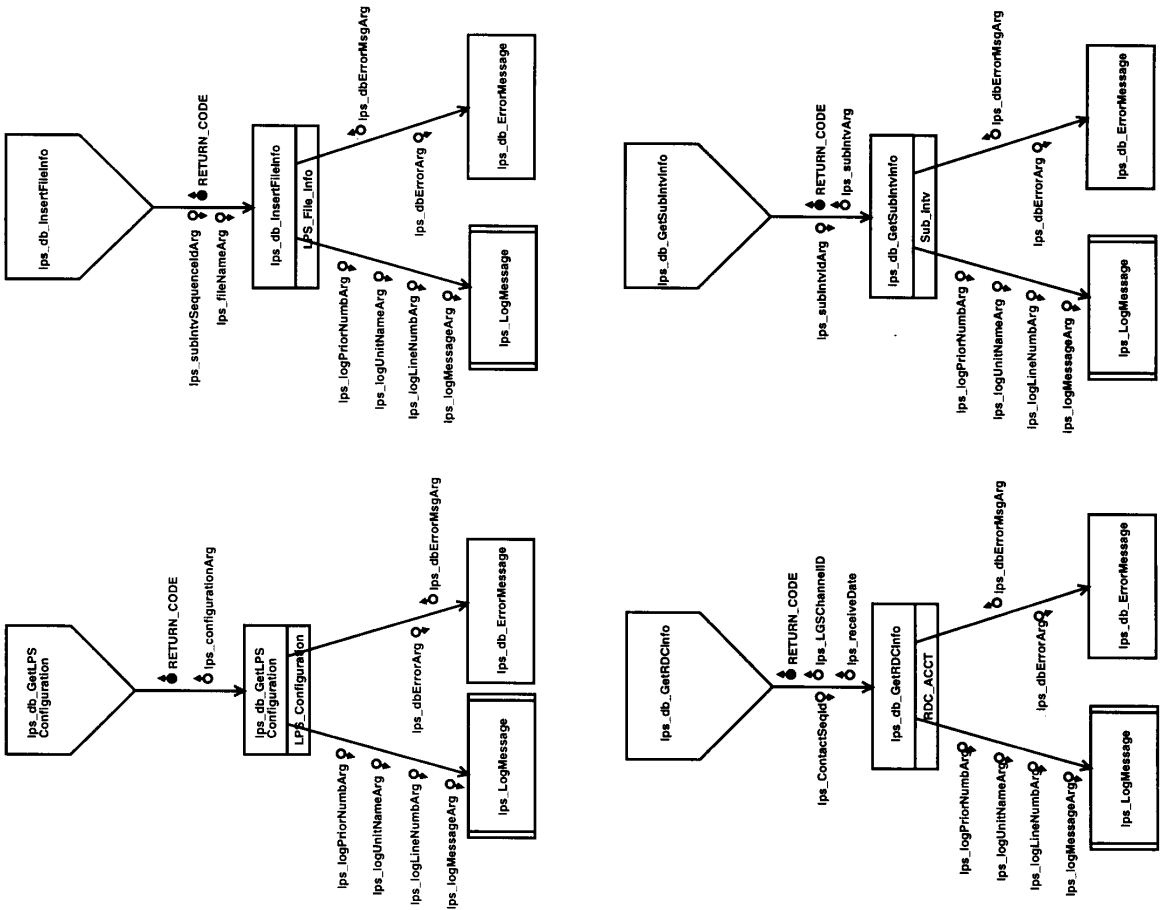


Figure 4–8. LPS Database Access Structure Chart (3 of 3)


lps_db_InsertFileInfo
RETURN_CODE
lps_subIntvSequenceIdArg
lps_fileNameArg
lps_db_InsertFileInfo
LPS_File_Info
lps_dbErrorMsgArg
lps_dbErrorArg
lps_db_ErrorMessage
lps_logPriorNumbArg
lps_logUnitNameArg
lps_logLineNumbArg
lps_logMessageArg
lps_LogMessage
lps_db_GetSubIntvInfo
RETURN_CODE
lps_subIntvIdArg
lps_db_GetSubIntvInfo
Sub_Intv
lps_dbErrorMsgArg
lps_dbErrorArg
lps_db_ErrorMessage
lps_logPriorNumbArg
lps_logUnitNameArg
lps_logLineNumbArg
lps_logMessageArg
lps_LogMessage
lps_db_GetLPS Configuration
RETURN_CODE
lps_configurationArg
lps_db_GetLPS Configuration
LPS_Configuration
lps_dbErrorMsgArg
lps_dbErrorArg
lps_db_ErrorMessage
lps_logPriorNumbArg
lps_logUnitNameArg
lps_logLineNumbArg
lps_logMessageArg
lps_LogMessage
lps_db_GetRDCInfo
RETURN_CODE
lps_LGSChannelID
lps_receiveDate
lps_ContactSeqId
lps_db_GetRDCInfo
RDC_ACCT
lps_dbErrorMsgArg
lps_dbErrorArg
lps_db_ErrorMessage
lps_logPriorNumbArg
lps_logUnitNameArg
lps_logLineNumbArg
lps_logMessageArg
lps_LogMessage

Section 5. Raw Data Capture Subsystem

5.1 Introduction

The RDCS (Figure 5–1) is responsible for capturing raw wideband data and storing it in a datastore for later processing by the RDPS. The RDCS design provides the capability to capture raw wideband data independent of the database's availability.

A captured raw wideband data set includes a raw wideband data file (a binary file) and a raw wideband accounting file (an accounting statistics file). The accounting file, an American Standard Code for Information Interchange (ASCII) text file, contains statistics information specific to a contact period for the captured raw wideband data. In the event that the database is unavailable during a capture session, this file is used to update the database at a later time.

A captured raw wideband data set for a contact period is saved onto removable media with a tape label generated for the saved raw wideband data file. The saved files are restaged for reprocessing, as requested, and deleted following a successful archive.

Throughout the capture process, the RDCS periodically logs status messages to monitor the health of the subsystem.

5.2 Design Overview

This section provides an overview of the RDCS software design. The relationship between the RDCS and the MACS is presented along with a discussion of the assumptions, constraints and considerations used in the design and implementation process.

5.2.1 Subsystem Software Overview

The RDCS interfaces with the MACS to receive a start and stop capture directive and to report status information. The RDCS is designed to execute via the graphical user interface (GUI) through the MACS or from the command line if the MACS and/or the database is unavailable. The MACS supplies the RDCS with the schedule identifier to start raw data capture. If the database and/or the MACS is unavailable, the user supplies the scheduled start capture time. The stop data capture occurs in one of four methods:

1. RDCS uses a MACS-supplied stop time argument.
2. RDCS uses a user-supplied capture stop time, if the database is not running.
3. RDCS uses a directive sent by the MACS to override the scheduled capture termination.
4. RDCS uses the default contact duration of 14 minutes, which ensures termination of the capture session.

Additionally, the capture process is nonpreempted, given the highest priority, and, optionally, isolated and assigned to a restricted processor until completion of a capture session. These options prevent interference during the capture session.

The RDCS captures the raw wideband data through the capture device, high-speed peripheral device interface (HPDI) board, to a disk file for RDPS processing. By default, during a capture

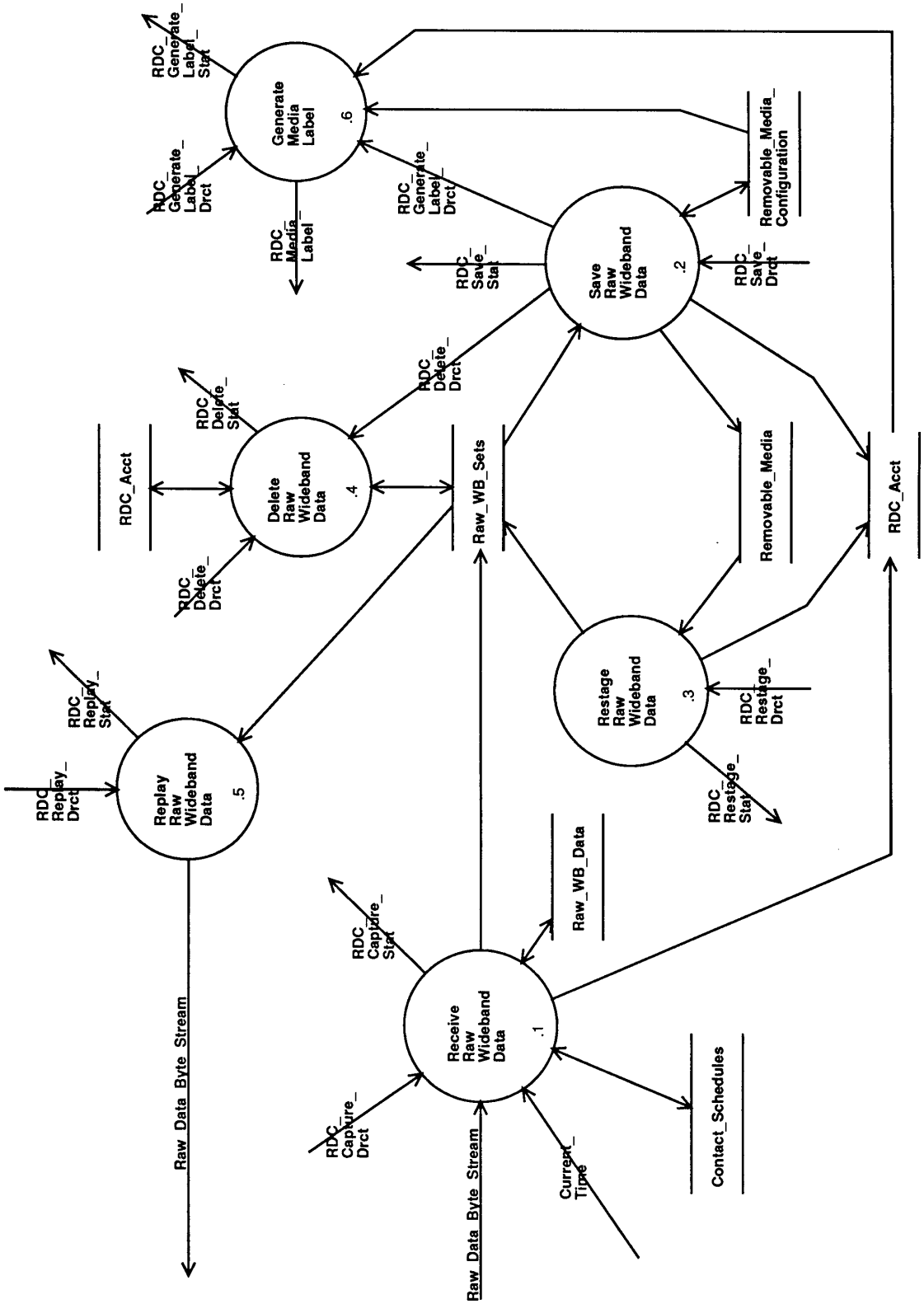


Figure 5–1. Raw Data Capture Context Diagram

session, all Level 0R processes are suspended until data capture is completed. Each captured file is uniquely named. The filename consists of the actual start time and date of the capture session, the capture source identifier, and an extension to distinguish between the raw wideband data and the accounting file.

Once the capture session completes, the RDCS saves (rdc_Save) the capture and associated accounting file to removable media for short-term storage as directed either through the MACS or the command line if the database is not running. Once saved, the archive_flag in the RDC_ACCT database table is set to true. rdc_Save generates a tape label for the removable media using data from the accounting file. If the database is currently unavailable, a tape label can be generated at a later time by executing rdc_GenLabel when the database is available. rdc_Save is not allowed to run if a capture is running.

Following a successful save, the files are removed from the capture disk based on two conditions: if it has been Level 0R processed or if it has been archived. If the database indicates that the two conditions are true, the files are deleted. Files can be deleted by executing rdc_DeleteFiles on the command line. In this case, the user must select the option of a conditional or unconditional delete. A conditional delete is based on the two conditions. An unconditional delete removes the capture files regardless of the two conditions. If the database is not running, both the raw wideband data file and its associated accounting file are not deleted.

In the event that the data requires reprocessing, the MACS sends a directive to the RDCS requesting a restage (rdc_Restage) or the restage process can be executed on the command line. The MACS supplies the device name where the files are to be extracted. If run on the command line, the user may specify the device name; otherwise, the default device name is used. When a capture file is restaged, the database is updated using the accounting file and the on_line_flag in the database is set to true. If the database is unavailable, restage will not occur.

To manually stop data capture, the user activates an executable from the command line (rdc_Terminate) if the database is not running or (if the database is running) through a MACS directive (GUI) that sends a signal to the RDCS to terminate the specified process.

For testing purposes, rdc_Transmit is a process that allows the transmission of data through the transmit device for capture by the capture device. During the capture, periodic messages are logged and/or displayed that allow the user to monitor the process.

5.2.2 Design Considerations

This subsection presents the design drivers relevant to the RDCS software design and implementation. The assumptions, software reuse strategy, and required operational support that influence the design of the RDCS software are described.

The main consideration in designing the RDCS is that raw data capture continues with or without all the requested resources, including the database and the MACS. Also, due to the nature of the architecture of the LPS, it is crucial that stop data capture occurs gracefully and does not require a “kill” command from the operator. Therefore, the design includes two options:

1. Default time is set to automatically terminate data capture after the maximum contact length has been reached should the manual or MACS directive termination process fail

2. rdc_Terminate executable to terminate the data capture

5.2.2.1Assumptions and Open Issues

The following assumptions are used in the RDCS software design and implementation:

- The RDCS captures data with or without a number of requested resources, including access to the database.
- To maintain independence, the raw data capture filename consists of easily accessible information without having to query the database.
- The RDCS will be allowed to capture raw data with or without the ability to collect and save accounting information.
- This design is for single processing. It does not allow for multiple capture, multiple saving to removable media, or multiple restaging of raw wideband data.
- There is only one contact file per removable medium.

There are no open issues.

5.2.2.2Operational Support

The RDCS is “forked” as a child process by the MACS. The RDCS is invoked from the command line if the database is not running.

- Status returns and error messages are logged into the LPS Journal file.
- Initialization begins by validating the arguments and recording the options selected. NOTE: The assumption here is that the global lps_LogMessage can be invoked with or without the database running.
- The capture disk is checked for space. If the disk is full, a message is displayed and a wait is initiated to allow the user to delete some of the files. If, after a predetermined duration, the capture disk does not have enough space for a full contact, the RDCS is terminated.
- If selected, all Level 0R processes running are suspended to avoid raw data receive interference. The capture process is nonpreempted and given the highest nondegrading priority, if so requested.
- A timer is set to the scheduled (or entered) stop capture time. At the stop time, if no data is being captured, capture device reset is activated to stop data receive. Resetting the capture device also can be activated by the user to manually stop data capture.
- Capture raw wideband data and accounting files are saved to removable media and restaged from removable media on request.
- The RDCS connects to the database if it is running and inserts the contact accounting information. A disk file is created containing the contact accounting information.

5.2.2.3Software Reuse Strategy

The RDCS will incorporate the LPS data capture prototype and software drivers developed by SGI for the HPDI board, Versa-Module European (VME) reset, and RAID storage devices.

5.3 Top-Level Models

rdc_Capture (Figure 5–2) captures the raw wideband data. lps_TimeGetCurrentTime obtains the current capture session's start time. rdc_Init initializes the capture session by processing the environment variables, providing default options in cases of unspecified arguments and connecting to the database, if available. rdc_CaptureInit validates the arguments provided to begin the capture session. Once the arguments have been successfully validated, rdc_CaptureData captures the raw wideband data. Once the capture session completes, rdc_CaptureCleanup cleans up and shuts down the capture process.

rdc_CaptureInit (Figure 5–3) initializes the capture session by validating the capture source, capture start time, and capture stop time arguments. Invalid arguments result in the use of the default arguments. rdc_SuspendProcess suspends the Level 0R processes on request. Additionally, rdc_IsolateProcess restricts the capture process to a specified processor and isolates the processor on request. The capture data filename and accounting filenames are created.

rdc_CaptureData (Figure 5–4) captures the raw wideband data. rdc_DevCaptureOpen opens, initializes, and resets the capture device. Upon success, rdc_DevCapture captures the raw wideband data via the capture device.

rdc_CaptureCleanup (Figure 5–5) calculates the capture raw wideband data statistics and writes the information to the accounting file, updates the database using the accounting file, and terminates the capture session. rdc_DevCaptureClose closes the capture device. rdc_CaptureCalcAccounting calculates the statistics from the capture session and stores the information into the accounting file using rdc_WritetoAcctFile. rdc_WriteAcctToDb updates the RDC_ACCT with the accounting information. rdc_ResumeProcess resumes suspended Level 0R processes. rdc_DeIsolateProcess unrestricts and deisolates the processor if restricted and isolated. New information recorded in the database is saved, and lps_db_Disconnect ends the database connection with rdc_Capture.

rdc_DeleteFiles (Figure 5–6) removes captured data files and the associated accounting file from the capture disk. The MACS or the user must provide the name of the file for deletion and the condition for deletion. lps_db_Connect connects rdc_DeleteFiles to the database. rdc_DeleteRDCFiles queries the database and deletes the files on the capture disk ,depending on the condition specified. If a file is marked for deletion and does not exist in the database, rdc_db_WriteAcctToDb updates the database with the new record using the accounting file. Additionally, rdc_db_WriteOnLineFlag sets the on_line_flag in the database table to indicate that the file either exists or does not exist on the capture disk. lps_db_Disconnect disconnects the process from the database on completion.

rdc_GenLabel (Figure 5–7) requires the raw wideband data filename to generate a tape label. rdc_GenLabel can be run independent of the MACS and rdc_Save, but can be forked either from the MACS or rdc_Save. lps_ProcessInit initializes the processing environment for the generate tape label process and connects to the database. rdc_GenLabel depends on the database for the tape

label information. rdc_db_LoadLabelParms queries the database for data necessary for tape label generation. Once the tape label information is available, rdc_PrintLabel prints the tape label information to the tape label on a dot matrix printer.

rdc_HpdiFunctions (Figure 5–8) provides the connection for the RDCS software to communicate with the hardware capture device, the HPDI board. The specified device is opened and reset in preparation for a transmit/capture session. All of the buffers are initialized. The data is transmitted to an LPS string through the HPDI board. The data is captured from the HPDI board to the capture disk.

rdc_Restage (Figure 5–9) extracts captured raw wideband data files previously saved on removable media. rdc_Init initializes the restage session by processing the environment variables, reading the capture options and providing defaults if the options are not provided, and connecting to the database, if available. If the database is not available, rdc_Restage terminates. rdc_db_RegisterProcess registers the restage process in the database as an indication of an active restage and unregisters the process on completion of restage. This signifies for the MACS whether the restage process is currently running. lps_db_Commit commits the database updates. rdc_LoadTape loads the removable media into the default tape drive. Before a restage occurs, rdc_CheckDiskSpace checks for the available disk space on the capture disk and uses the current system time obtained from lps_TimeGetCurrentTime to determine status reporting intervals. rdc_SystemMonitor executes the “tar” command to extract raw wideband data files from the removable media. The tar command generates an output file, rdc_ReadTarOutputFile, listing the contents on the removable media. This file is read and rdc_db_FileExistence determines if the files exists in the database. If the file currently exists in the database, rdc_db_WriteOnLineFlag sets the on_line_flag in RDC_ACCT to true. If the file does not exist in the database, rdc_db_WriteAcctToDb updates the database using the accounting file. rdc_UnLoadTape unloads the tape from the tape device, and lps_db_Disconnect disconnects the restage session from the database.

rdc_Save (Figure 5–10) saves raw wideband data files to removable media. rdc_Init initializes the save session by processing the environment variables, reading the capture options and providing defaults if the options are not provided, and connecting to the database, if available. rdc_db_RegisterProcess registers the save process and unregisters the save process on completion. This indicates to the MACS that rdc_Save is currently running. rdc_SaveValidateArgs validates all of the arguments provided and uses default values for invalid arguments. rdc_TapeBinCount determines the total number of tape slots available on the tape device. rdc_GetBinNumber reads the lpsTapeLibraryBinFile for the current tape slot number to use for archiving. rdc_SetBinNumber updates the lpsTapeLibraryBinFile with the tape slot number using. rdc_LoadTape automatically loads the tape into the tape slot number specified in the lpsTapeLibraryBinFile for archiving. rdc_FileSplit parses the raw wideband data filename provided into the path and filename. lps_TimeGetCurrentTime obtains the current system time and used in conjunction with rdc_TimeDiff to determine the status reporting at a specified interval. rdc_SystemMonitor executes the tar command that saves the raw wideband data to tape. Once the files have been successfully saved to removable media, rdc_db_SetArchiveFlag sets the archive_flag in RDC_ACCT to true. Raw wideband data files are then deleted using rdc_DeleteFiles. lps_ProcessStartChild then forks off the rdc_GenLabel process to generate a tape label for the saved raw wideband data.

rdc_Terminate (Figure 5–11) terminates any RDCS process.

rdc_UpdRDCAcct (Figure 5–12) updates the RDC_ACCT table. This process must first connect to the database by lps_db_Connect. Using rdc_System determines if any accounting files exist in the capture directory. If the capture directory is unavailable, the current directory is checked. If accounting files exists in the capture directory and rdc_db_FileExistence indicates the file does not exist in the database, rdc_db_WriteAcctToDb updates RDC_ACCT with the accounting information. If rdc_db_FileExistence indicates that a file does exist in the database, no updates occur. rdc_db_SetOnLineFlag sets each on_line_flag in RDC_ACCT to true. On completion, lps_db_Disconnect is called to disconnect the update session from the database.

rdc_Transmit (Figure 5–13) is provided to transmit raw wideband data to the capture process for testing purposes. rdc_Init initializes the save session by processing the environment variables, reading the capture options and providing defaults if the options are not provided, and connecting to the database, if available. rdc_TransmitInit initializes the transmit session. rdc_TransmitData begins data transmission. On completion of the data transmit, rdc_TransmitCleanup cleans up and shuts down the transmit session.

rdc_TransmitInit (Figure 5–14) validates the transmit arguments by calling rdc_TransmitValidateArgs, which validates the filename, determines whether or not the file exists, and determines if the file is accessible and has sufficient read privileges. rdc_SuspendProcess suspends the Level 0R process on request. rdc_IsolateProcess isolates and restricts the transmit process to a specified processor upon request.

rdc_TransmitData (Figure 5–15) transmits the raw wideband data through the transmit device. A status reporting interval is initiated by determining the current time. rdc_DevTransmitOpen opens the transmit device and begins data transmission through the rdc_DevTransmit routine.

rdc_TransmitCleanup (Figure 5–16) gracefully ends the transmit session. rdc_ResumeProcess resumes any suspended Level 0R processes. rdc_DeIsolateProcess deisolates and unrestricts a process from a specified processor if the process was previously isolated and restricted. The transmit is closed using rdc_DevTransmitClose.

All errors are logged into the LPS Journal file.

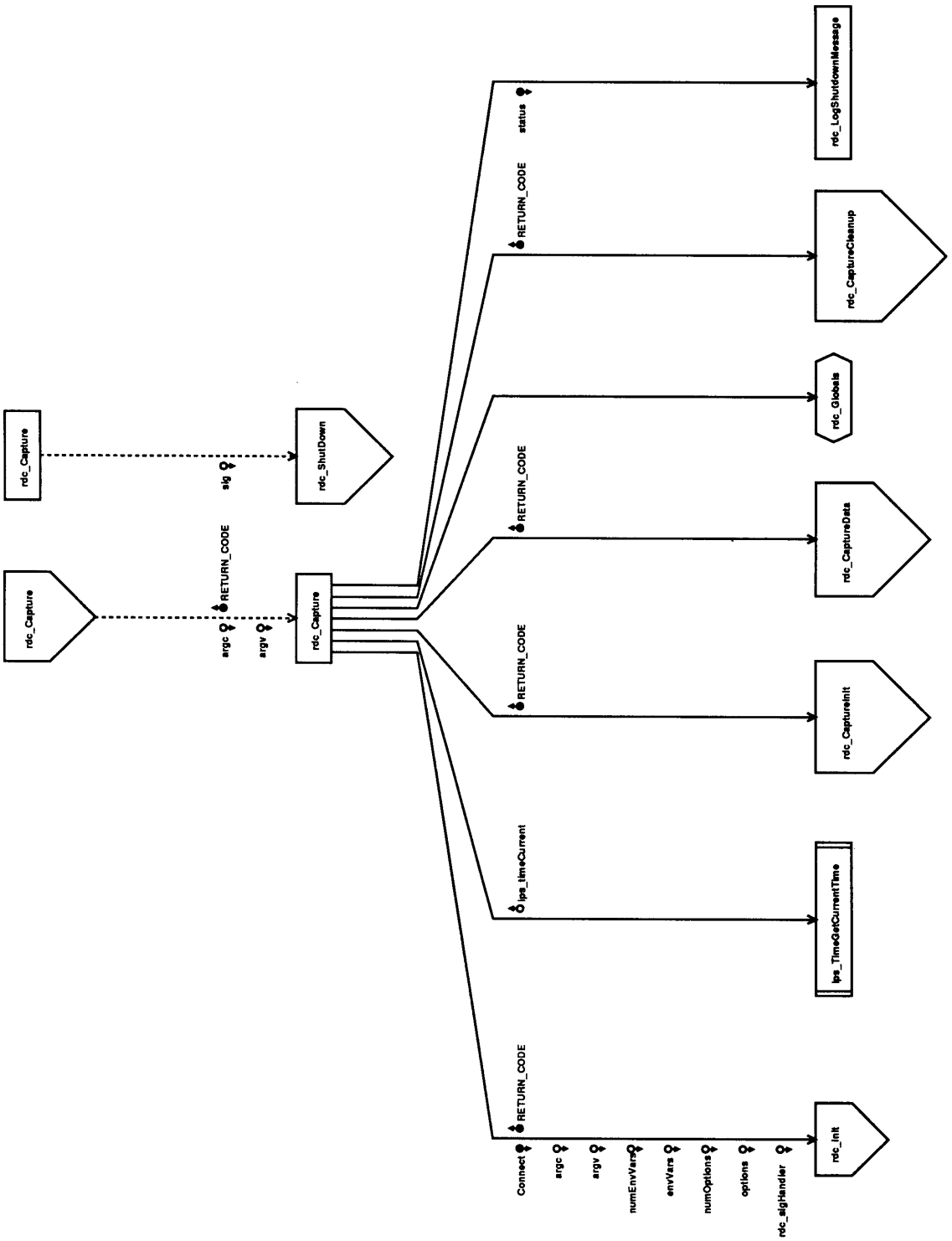


Figure 5–2. rdc_Capture Structure Chart

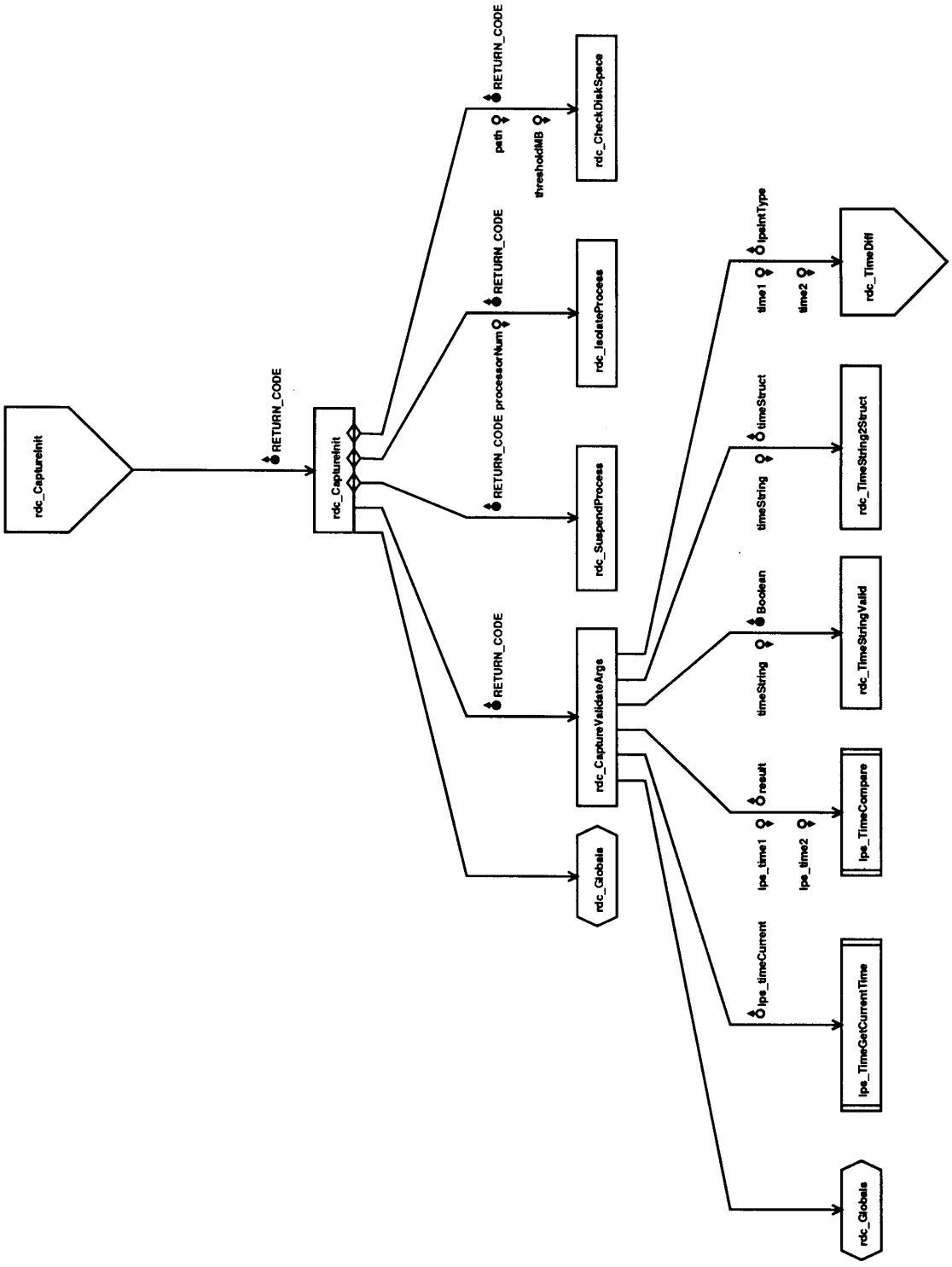


Figure 5–3. rdc_CaptureInit Structure Chart

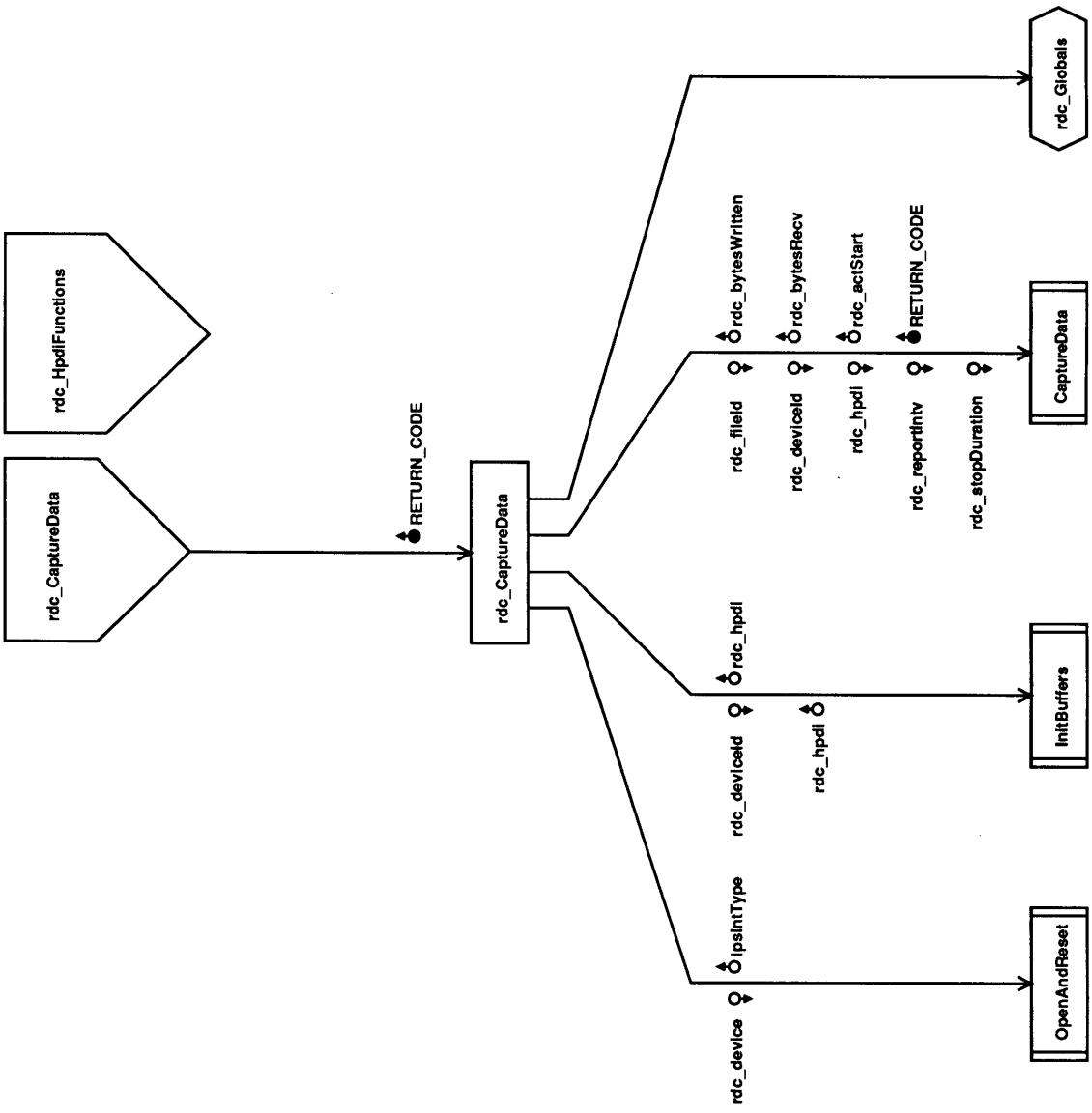


Figure 5–4. rdc_CaptureData Structure Chart

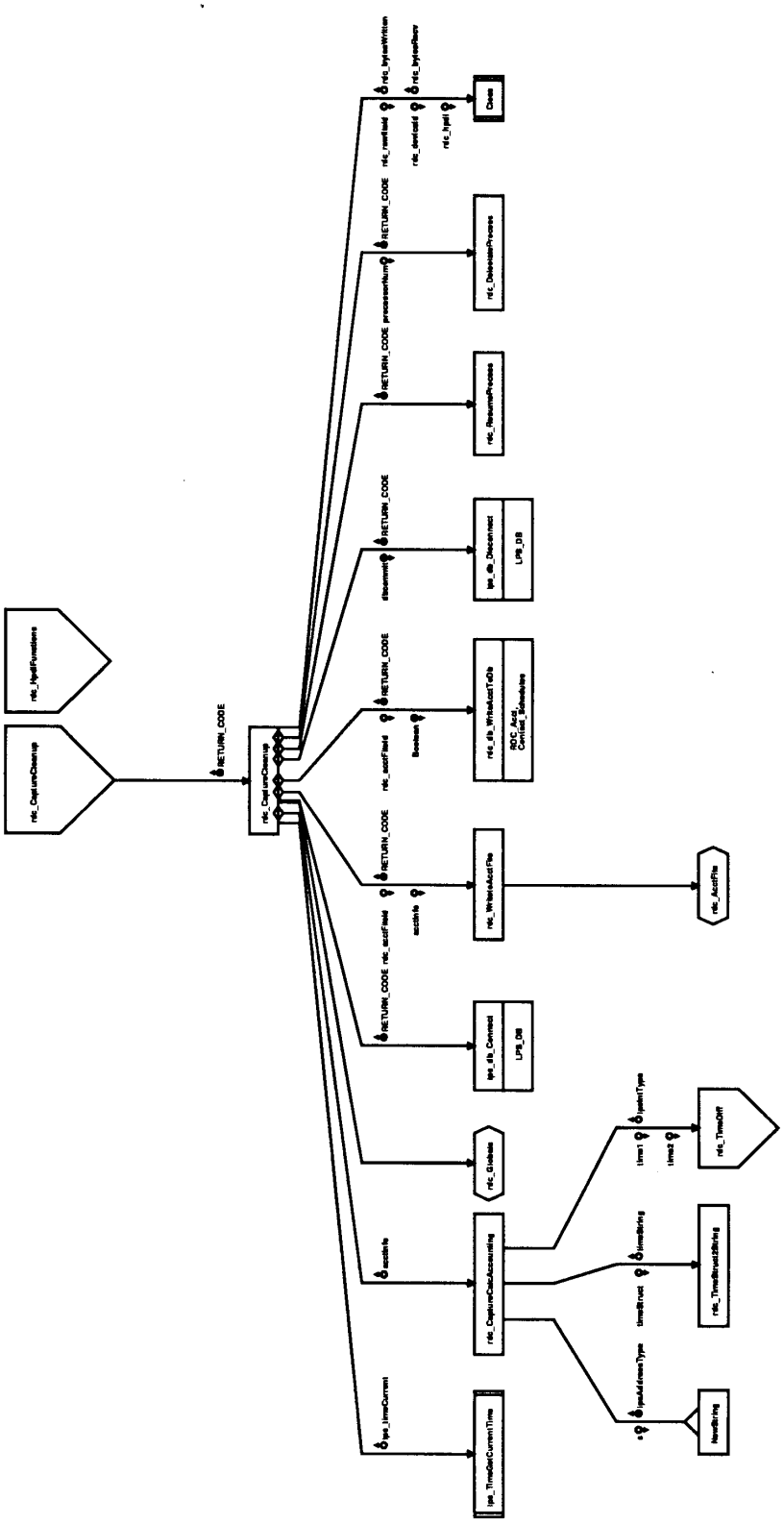


Figure 5–5. rdc_CaptureCleanup Structure Chart

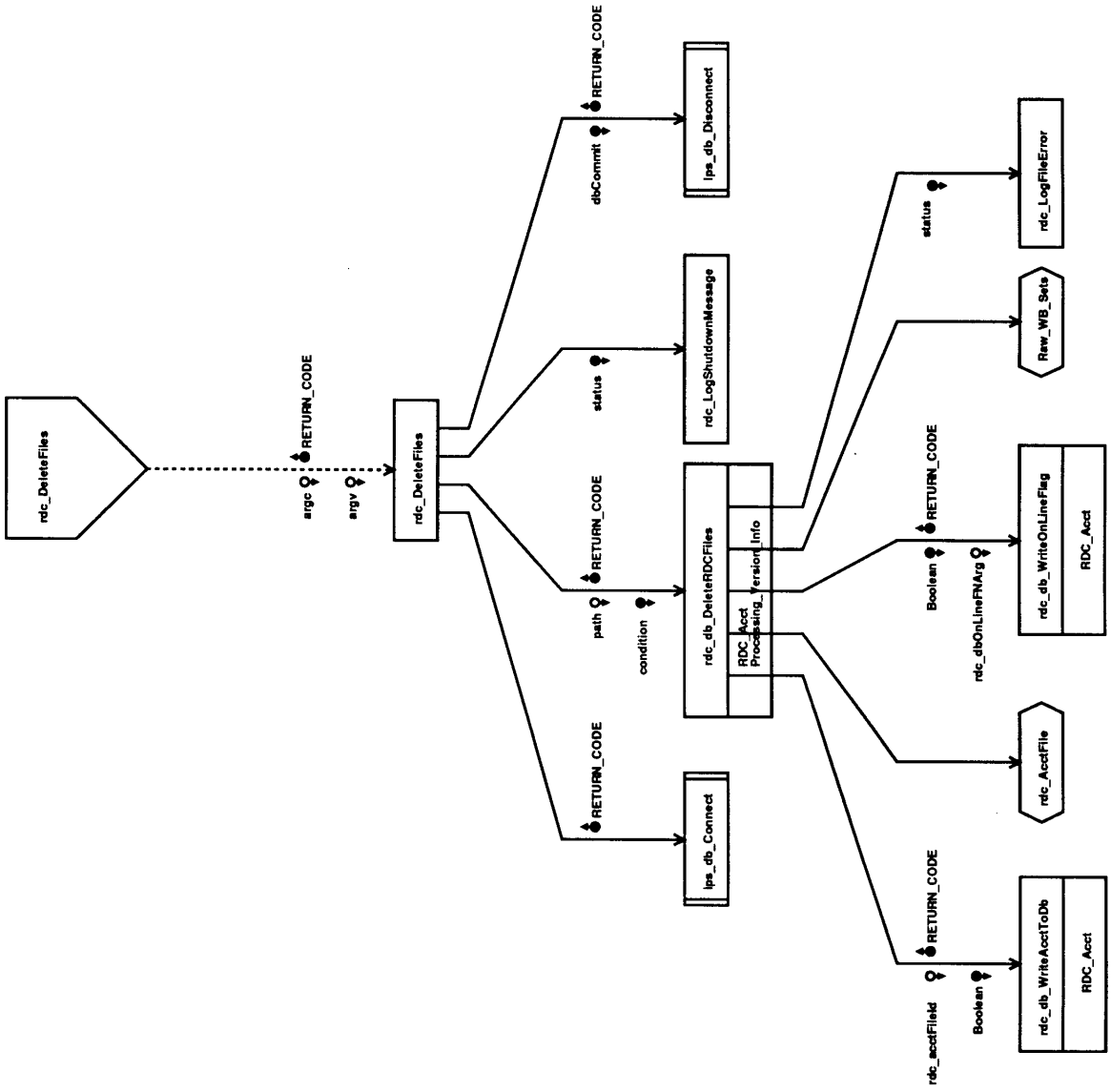


Figure 5–6. rdc_DeleteFiles Structure Chart

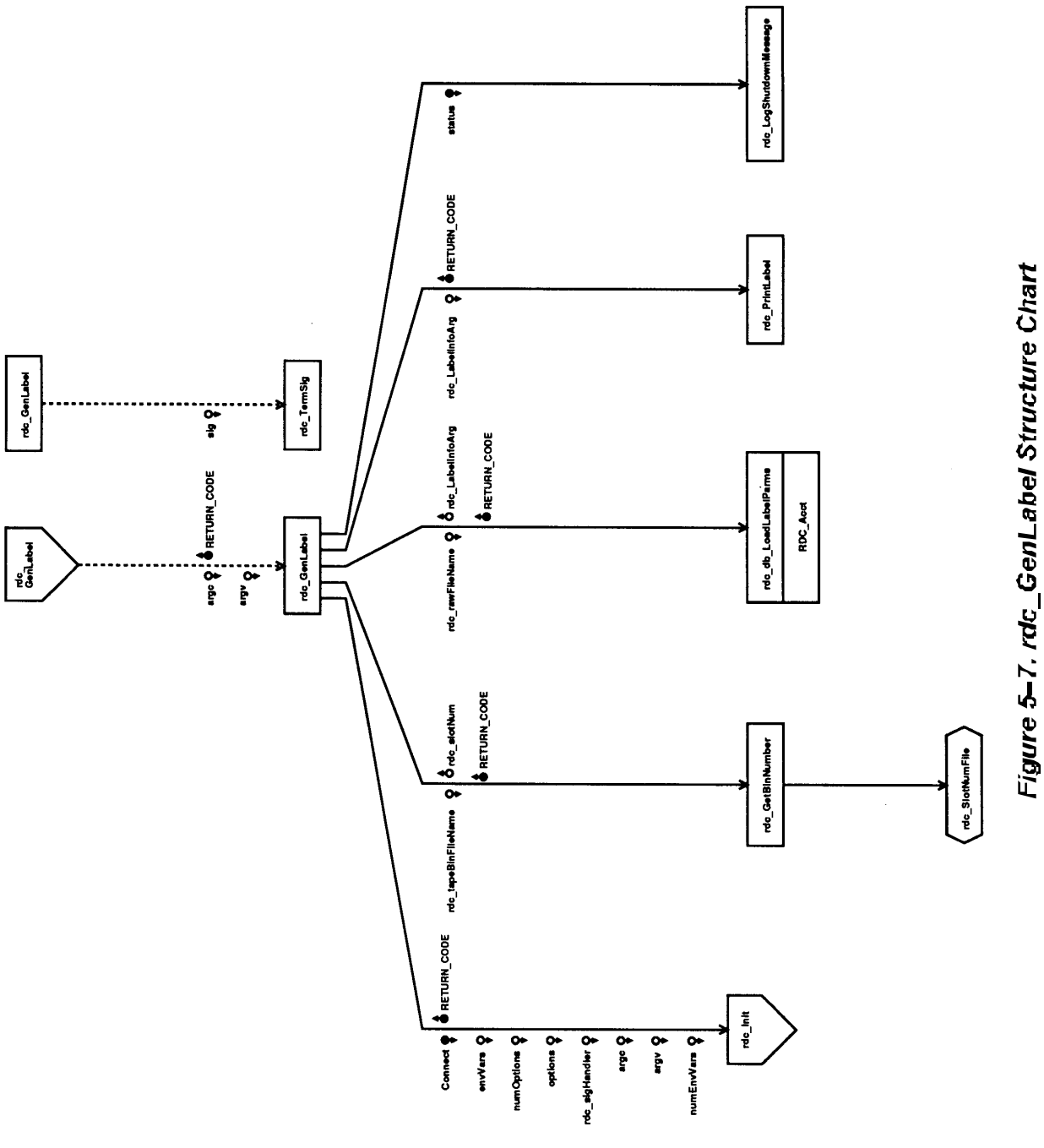


Figure 5–7. rdc_GenLabel Structure Chart

Figure 5–8. rdc_HpdiFunctions Structure Chart

Figure 5–9. rdc_Restage Structure Chart

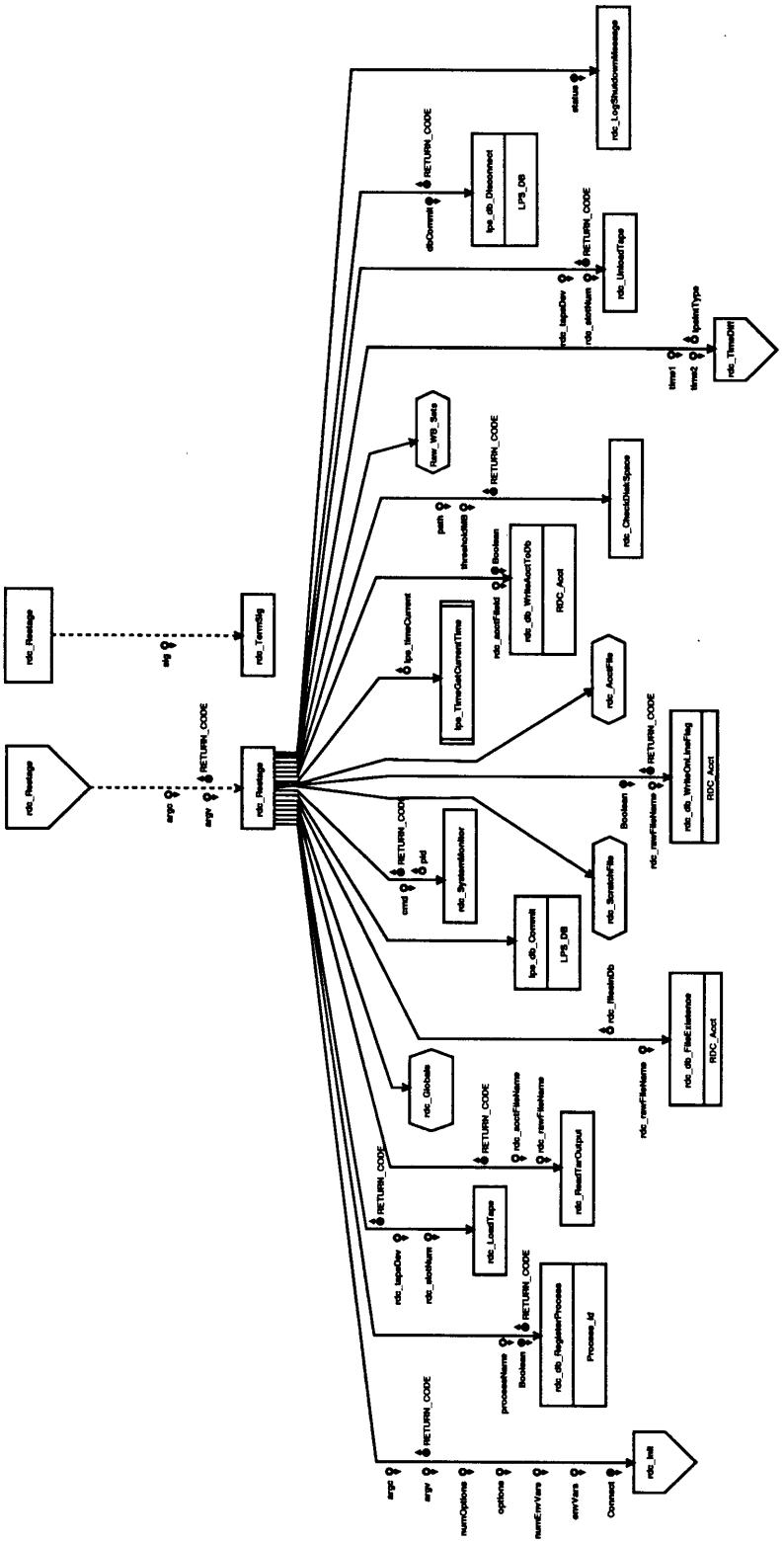


Figure 5–10. rdc_Save Structure Chart

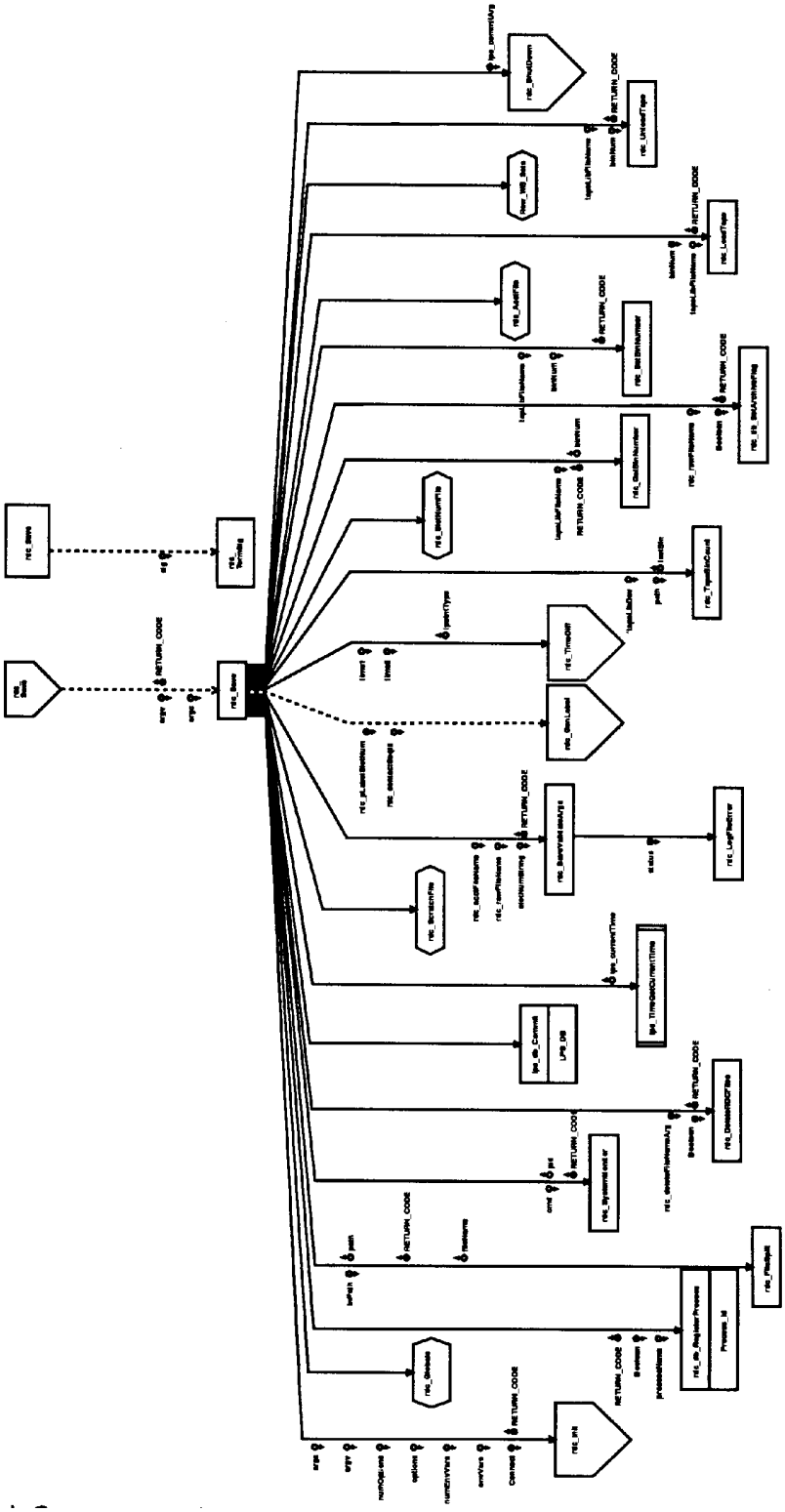


Figure 5–11. rdc_Terminate Structure Chart

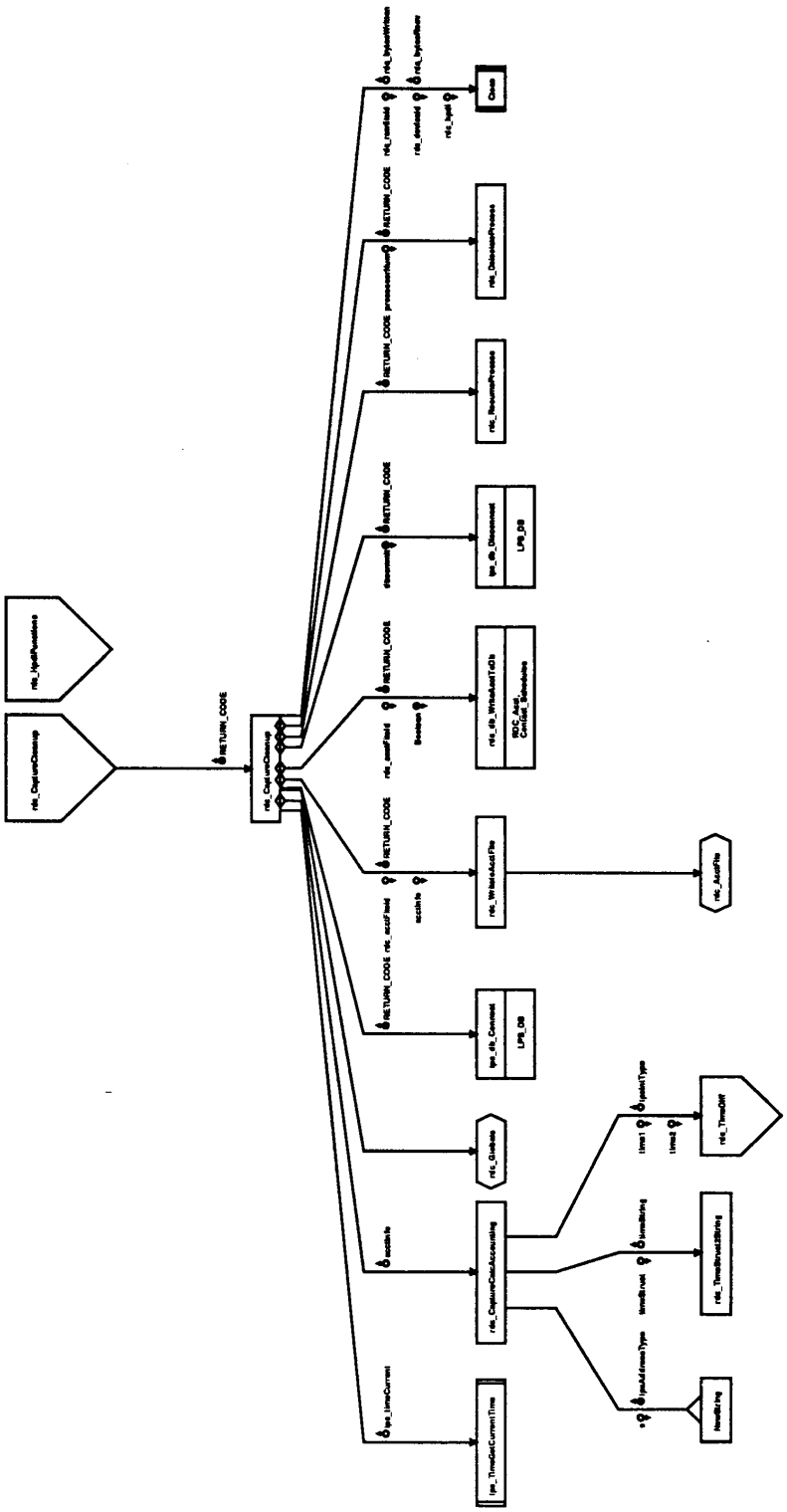


Figure 5–12. rdc_UpdRDCAcct Structure Chart

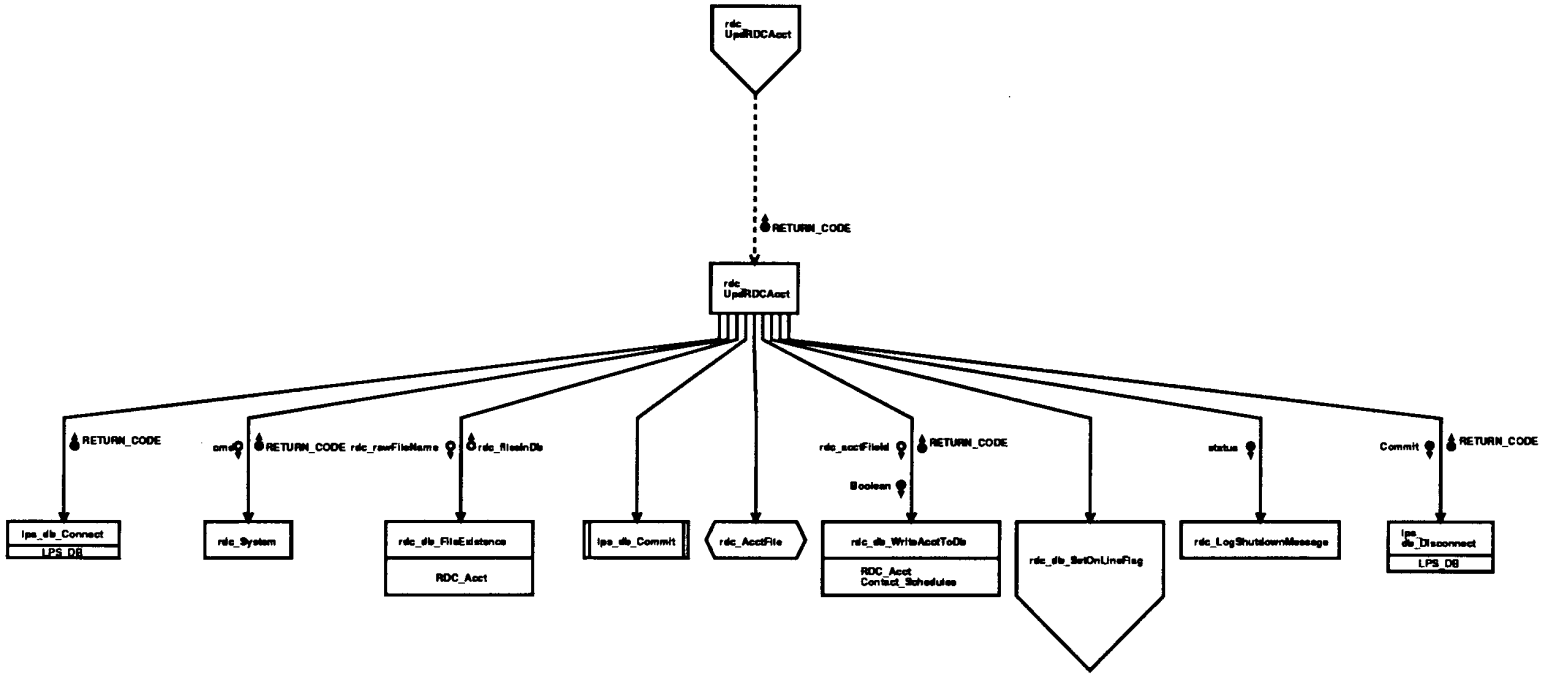


Figure 5–13. rdc_Transmit Structure Chart

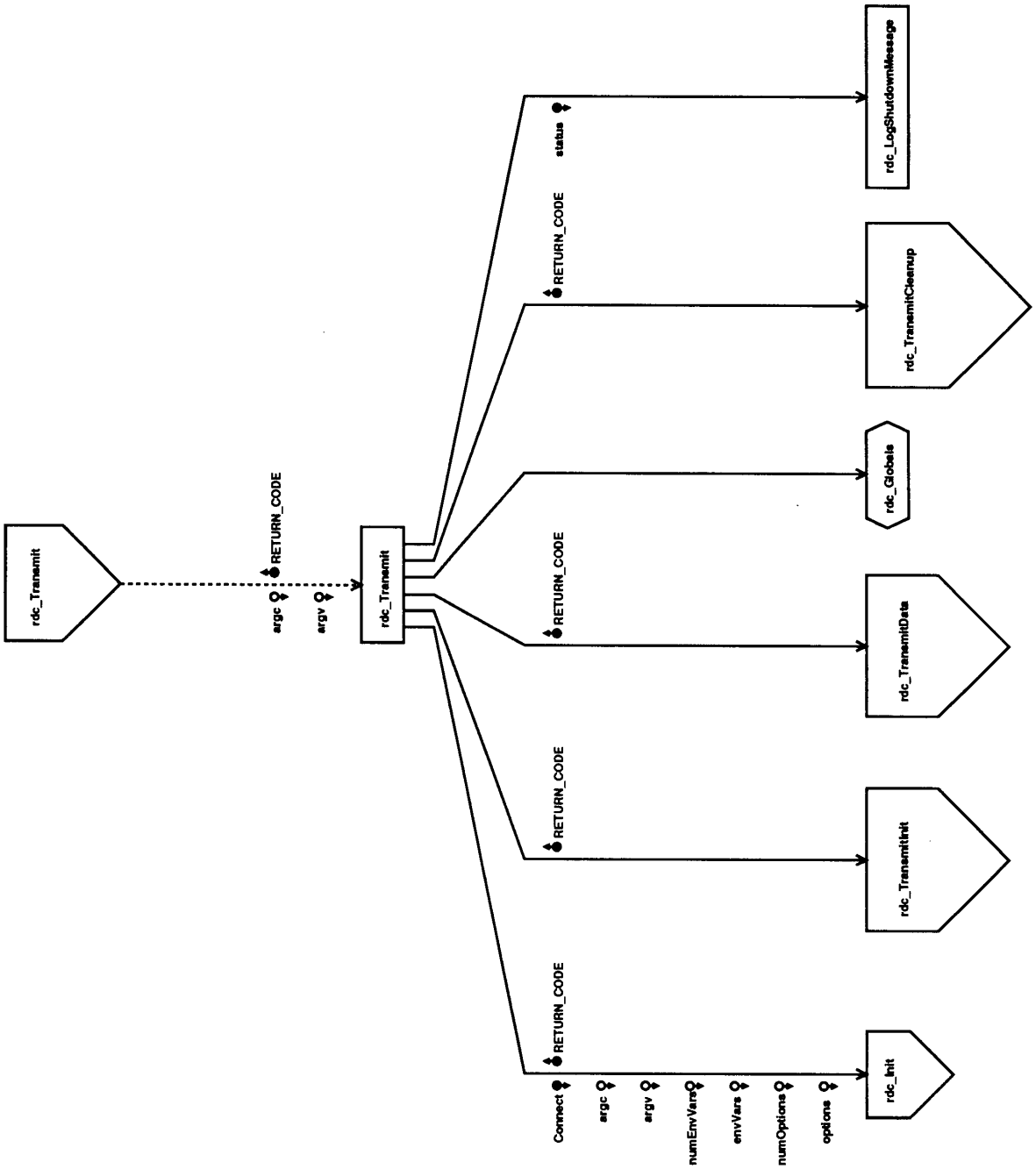
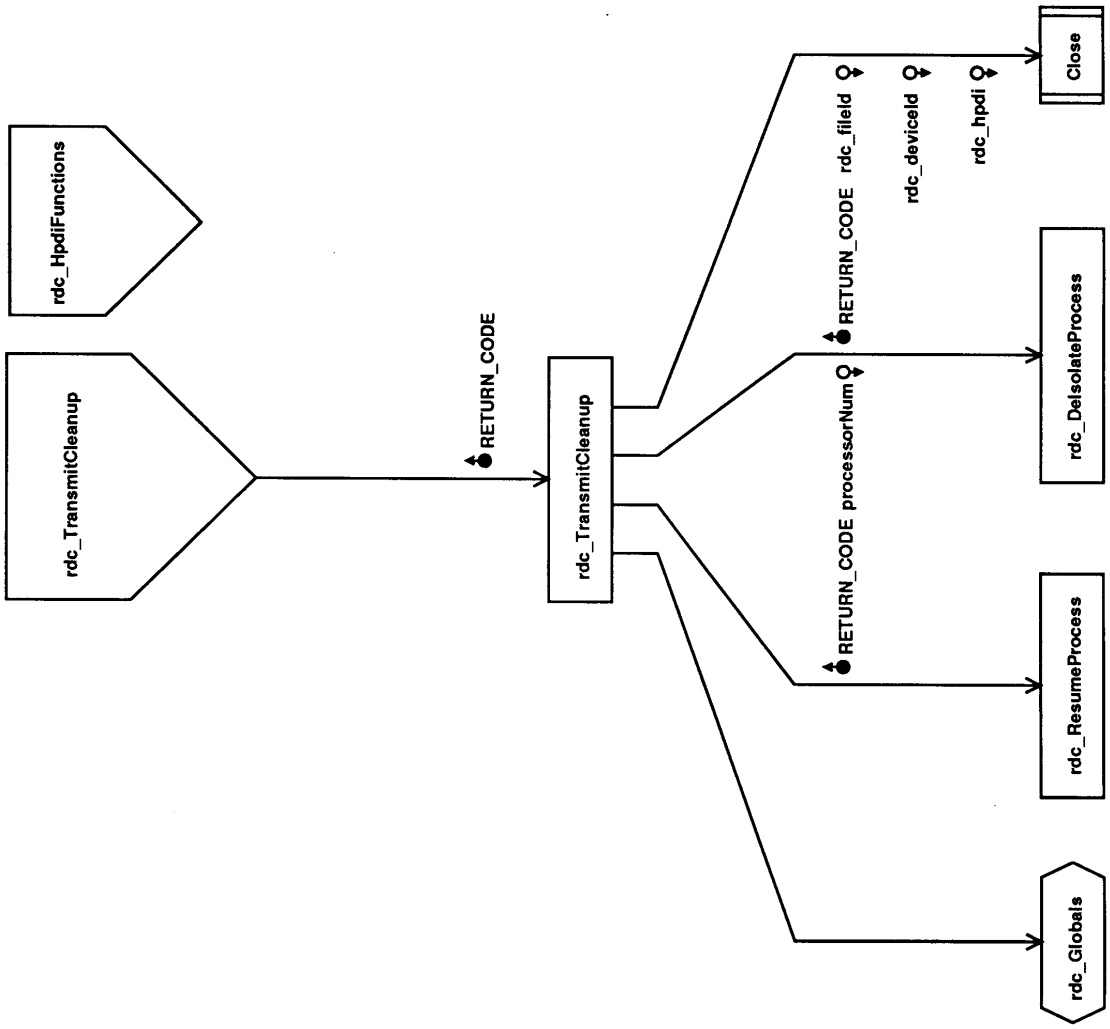


Figure 5–14. rdc_TransmitInit Structure Chart

Figure 5–15. rdc_TransmitData Structure Chart

Figure 5–16. rdc_TransmitCleanup Structure Chart


rdc_HpdiFunctions
rdc_TransmitCleanup
RETURN_CODE
rdc_TransmitCleanup
RETURN_CODE rdc_fileId
rdc_deviceId
rdc_hpdi
Close
RETURN_CODE processorNum
rdc_DeIsolateProcess
rdc_ResumeProcess
rdc_Globals

Section 6. Raw Data Processing Subsystem

6.1 Introduction

The RDPS is responsible for retrieving raw wideband data for the contact period specified by the operator. It synchronizes and annotates the channel access data units (CADUs) contained in the raw wideband data, performs error detection, and, to some degree, corrects errors in the data and makes the CADUs available to the MFPS.

The RDPS provides the quality and end of contact indicators to then MFPS and generates the Q&A information to be used in generating the return-link Q&A report. It also saves the CADUs that have failed the cyclic redundancy check (CRC), Reed-Solomon (RS), and Bose-Chaudhuri-Hocquenghem (BCH) error detection in a failed CADUs file by contact period.

6.2 Design Overview

This section provides an overview of the RDPS software design. The relationship between the RDPS and the other Landsat 7 subsystems is presented with a discussion of the assumptions, constraints, and considerations used in the design process.

6.2.1 Subsystem Software Overview

The RDPS interfaces with the RDCS to retrieve the raw wideband data that has been received and stored to the disk by the RDCS. The RDPS also interfaces with the MFPS to make the CADUs and associated quality information available as the RDPS completes processing a group of CADUs, referred to as a block. In addition, the RDPS interfaces with the MACS to receive information about the raw data file to process. It also retrieves the valid CCSDS parameters and error thresholds from the database and stores the Q&A information for the return-link Q&A report to the database. The context diagram is shown in Figure 6–1.

The RDPS retrieves blocks of data from the raw wideband data file whose filename was provided by RDCS in the database. The then RDPS searches for valid frame synchronization markers, identifies the CADUs, inverts them when necessary, and pseudorandom noise (PRN) decodes the CADUs. The CADUs are then stored in shared memory for subsequent processing by the RDPS and the MFPS. CADUs that have synchronization errors but can be identified as CADUs are flagged. Some CADUs are allowed to pass through despite the fact that the number of erroneous bits in the sync pattern exceeds the error tolerance. Such CADUs are referred to as flywheel CADUs. These CADUs are appropriately flagged.

Once CADUs have been identified, they are checked for CRC and RS errors. If no more than two RS errors are discovered, they are corrected. CADUs with CRC errors are flagged, and those with correctable and uncorrectable RS errors are identified. CADUs with fill data are flagged to be omitted during MFPS processing. Next, BCH error detection and correction is performed on the CADUs if there are bit slips, frame synchronization errors, or CRC errors or if errors have been detected by the RS algorithm. In addition, there is a provision to perform BCH error detection and correction on all CADUs if an appropriate flag is set by the invoking process. Separate BCH error detection and correction is provided for the mission data zone, as well as the data pointer. The mission data zone, together with its BCH zone, is 8-way interleaved, giving rise

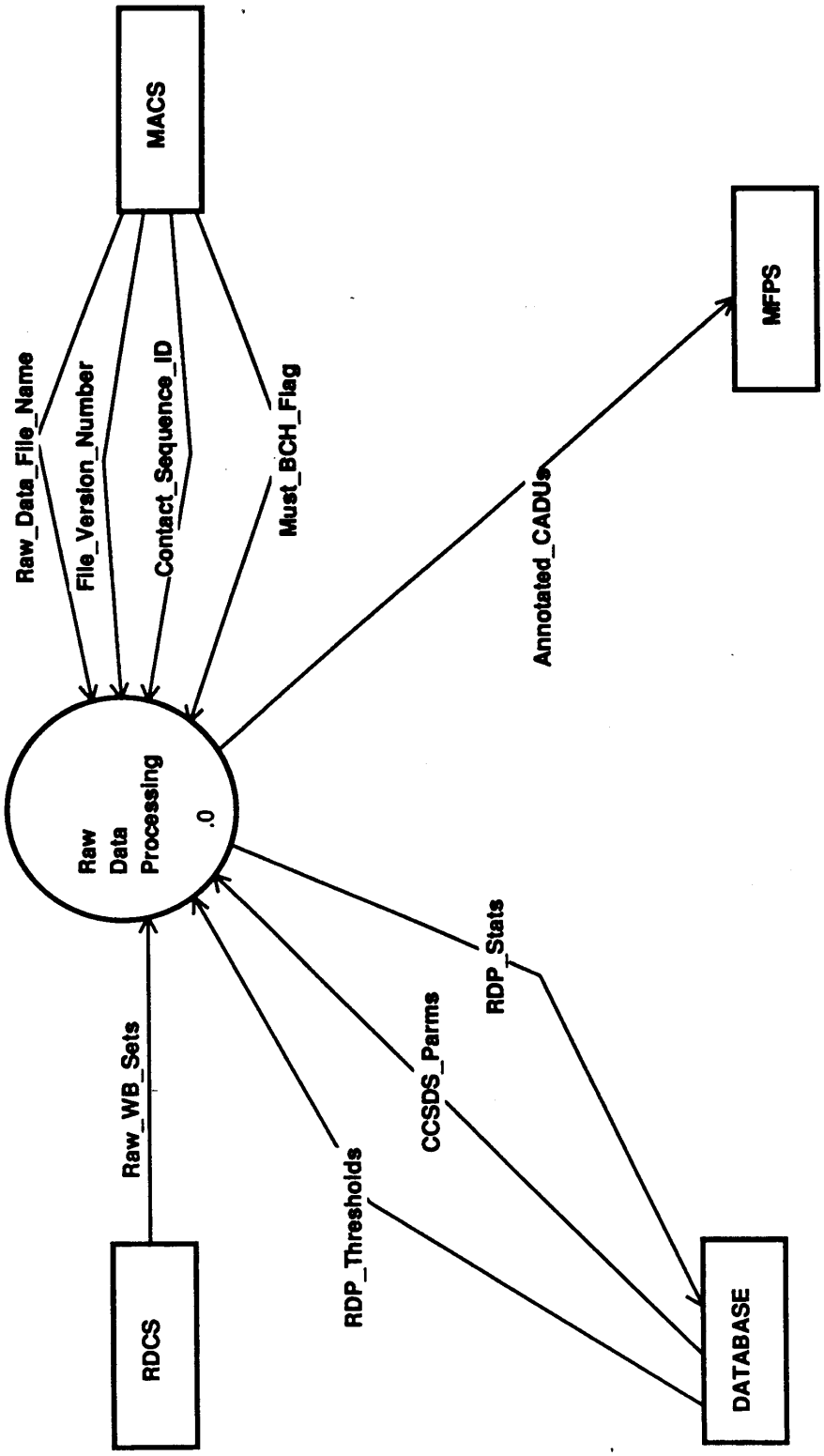


Figure 6–1. Raw Data Processing Context Diagram

to eight BCH code blocks. The pointer area, together with its BCH zone, is just one code block. Thus, there are nine distinct code blocks. BCH can correct up to three bit errors per code block for a maximum of up to 27 corrected bits for all of the code blocks. CADUs that have been BCH corrected are identified, as are those with BCH errors that could not be corrected.

The Q&A statistics for the return-link Q&A report are calculated and checked against the predetermined error thresholds. If the thresholds have been exceeded, a warning message will be logged. The CADUs that have failed the CRC, RS, or BCH error detection processes are written to a trouble file, along with their annotations. One of the annotations is the CADU's location within the raw wideband data file. When a block of shared memory has been processed, the shared memory area containing the CADUs is released to the MFPS.

6.2.2 Design Considerations

This section presents the design drivers relevant to the RDPS software design. The assumptions, software reuse strategy, and required operational support that influence the design of the RDPS software are described.

6.2.2.1Assumptions and Open Issues

The following assumptions have been made by the RDPS:

- The raw wideband data for the requested contact period has been captured by the RDCS and has been stored in a file with the filename that has been made available to the RDPS.
- BCH will be performed only if there are any bit slips or frame synchronization, CRC, or RS errors, unless a flag is set by the user to always perform BCH error detection and correction.
- The RDPS will identify fill CADUs to be ignored by the MFPS.

All of the open issues for the RDPS have been resolved.

6.2.2.2Operational Support

The RDPS will be forked by the MACS passing the contact sequence identifier and file version number of the data to be processed by the RDPS, a flag to indicate if BCH should always be performed, and the raw data filename. The RDPS performs the following operations:

- Connect to the database to allow for retrieval and insertions.
- Create shared memory areas for storing the CADUs and annotations to be shared by the RDPS and the MFPS.
- Read the raw wideband data from disk for a specified contact period.
- Perform frame synchronization on the data using a search, check, lock, and flywheel (SCLF) strategy.
- Synchronize data with normal and inverted polarity.

- Perform PRN decoding on the data.
- Perform CCSDS acquisition of signal (AOS) grade 3 checks, including CRC and RS.
- Continue processing of raw wideband data, even if errors are detected in the data or in event of a flywheel CADU, unless frame synchronization fails.
- Identify the CADUs containing fill data to be ignored by the MFPS.
- Perform BCH error detection and correction.
- Save CADUs that have failed CRC, RS, and BCH checks to a trouble file.
- Identify a change in the virtual channel identifier (VCID).
- Provide annotations for each CADU.
- Store return-link Q&A data to the database to be used for the return-link Q&A report.
- Log status and error messages.
- Set a signal trap to catch any signals that will cause abnormal termination.

6.2.2.3 Software Reuse Strategy

This section identifies the external components that may be reused by the RDPS, as well as common components of the RDPS software that may be useful to other Landsat 7 subsystems. Table 6–1 lists the component type and the ease of use classification for each reusable component.

Table 6–1. Reusable Components

Reusable Component	Type	Ease of Use
fs-frame-sync Component to perform frame synchronization on raw wideband data using the SCLF strategy	Design/code	No modifications
rdp_CRCCheck Component used to perform CRCs on raw wideband data.	Design/code	Minor modifications required
rsd (to be provided by Steve Duran) Component to perform RS error detection and correction on the header of the raw wideband data	Design/code	No modifications
rdp_BCHDecode Component to perform BCH error detection and correction on raw wideband data	Design/code	Major modifications required

6.2.3 Subsystem Error Handling

This section addresses the errors and processing exceptions that may be encountered by the RDPS. Table 6–2 describes the system response to each of the errors and the severity classification of each error.

Table 6–2. Errors and Their Severity

Error	Severity	System Response
Failure to connect to database	AL/HALT/PROC	If the RDPS is unable to connect to the database, processing for the contact period will be discontinued. Error messages will be logged.
Failure to retrieve CCSDS parameters	AL/HALT/PROC	The RDPS will terminate if the CCSDS parameters cannot be retrieved. Error messages will be logged.
Failure to retrieve raw data processing thresholds	WARNING	The default value will be used if the raw data processing thresholds are not available. Warning messages will be logged.
Failure to open or read Raw_WB_Sets file	AL/HALT/PROC	Processing for the RDPS will be discontinued if the file cannot be accessed. Error messages will be logged.
Failure to create or open shared memory areas	AL/HALT/PROC	Processing will be discontinued for the RDPS if the shared memory areas are not available. Error messages will be logged.
Failure to open or write failed CADUs file	WARNING	A warning message will be logged.
Failure to open or read BCH quad file	AL/HALT/PROC	Processing for the RDPS will be discontinued if the file cannot be accessed. Error messages will be logged.
Raw data processing threshold values exceeded	NOTIFY	If the synchronization, RS, CRC, or BCH errors exceed the thresholds, a message will be logged to be sent to the operator.
Data capture is running	AL/HALT/PROC	The RDPS will be terminated if data capture is running.

6.3 Subsystem Design

6.3.1 Top-Level Model

The top-level structure chart for the RDPS is shown in Figure 6–2. The driver of the RDPS software is rdp_Main. rdp_MainInit is called initially to connect to the database, obtain database information, set up the termination signal handler, and initialize files and shared memory areas. The CADUs are extracted from the raw wideband data file, identified, and written to shared memory in rdp_MainExtractCADU. CCSDS Grade 3 error detection and correction, including CRC and RS, is performed in rdp_MainValidateCADU. BCH error detection and correction occurs in rdp_BCHDecode. The failed CADUs file is generated in rdp_MainGenerateOutput. The shared memory is released here also to allow for accessing by the MFPS. The accounting information for the contact period is then stored to the database. Finally, the RDPS disconnects from the database, closes files, and detaches from shared memory in rdp_MainShutdown.

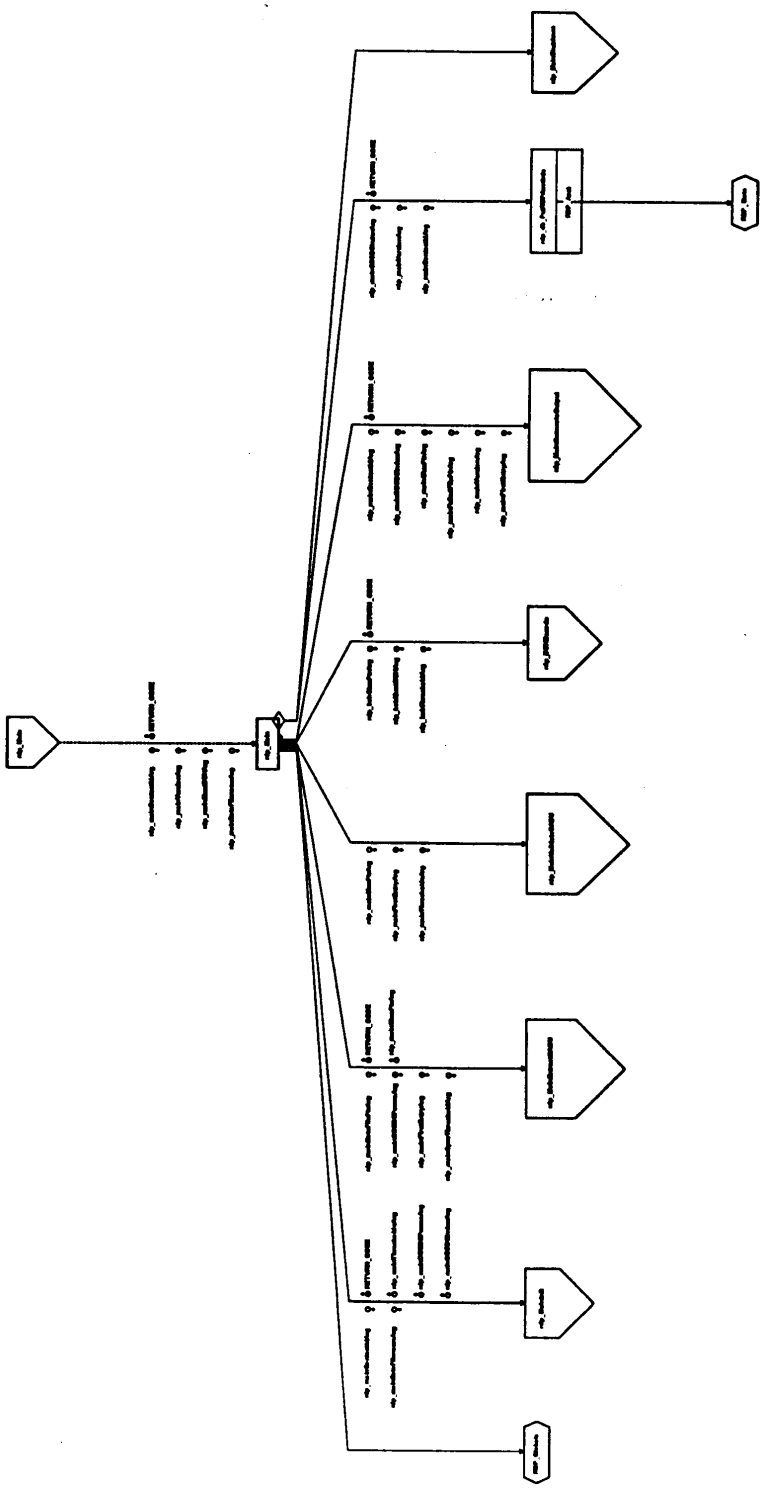


Figure 6–2. rdp_Main Structure Chart

6.3.2 Detailed Module Design

6.3.2.1rdp_Main

The rdp_Main module calls rdp_MainInit to access the database, open files, and set up the shared memory areas. For each block of raw wideband data for the specified contact period, rdp_Main calls rdp_MainExtractCADU to extract the data and identify blocks of CADUs, rdp_MainValidateCADU to use CRC and RS algorithms to check for errors, rdp_BCHDecode to perform BCH error detection and correction on the CADUs, and rdp_MainGenerateOutput to prepare the raw data processing output. After all blocks for the contact period have been processed, rdp_db_PutRDPAcctInfo stores the Q&A statistics to the database, and rdp_MainShutdown is called to clean up the RDPS.

6.3.2.2rdp_MainInit

The RDPS is initialized by rdp_MainInit (Figure 6–3). rdp_MainInit first defines the signal handling routines and connects to the database by calling lps_ProcessInit. rdp_db_GetCCSDSParms retrieves the CCSDS parameters from the database, and rdp_db_GetThresholds obtains the raw data processing error thresholds. The file containing the raw wideband data and the file for storing the failed CADUs are opened. The last function is to call lps_ShmOpen to open the shared memory area to store the annotated CADUs.

6.3.2.3rdp_MainExtractCADU

rdp_MainExtractCADU (Figure 6–4) first calls rdp_MainObtainData to retrieve a block of CADUs from the input raw wideband data file. This procedure is not done if space is not available in the shared memory because annotated CADUs that were previously placed into shared memory by the RDPS have not yet been processed by the MFPS. In this case, the RDPS will wait until the MFPS has caught up. The rdp_MainFSync subroutine (Figure 6–5) is then called to identify the CADUs using an SCLF strategy. It then stores the CADUs in shared memory along with the frame synchronization annotations. As CADUs are identified, they are normalized. If a CADU is reversed, the software reverses the CADU and removes the PRN encoding. If a CADU is inverted, the software inverts the CADU and removes the PRN encoding. The frame synchronization error indications and end of contact period flag are stored in the annotations. Error statistics regarding synchronization errors, bit slips, and flywheel CADUs are tallied. If a predetermined synchronization error threshold is exceeded, a message is logged in the Journal file.

6.3.2.4rdp_MainValidateCADU

CCSDS grade 3 processing is controlled in rdp_MainValidateCADU (Figure 6–6). The CADUs are validated one block at a time. First, CRCs are performed using the checksum calculations and a lookup table. RS error detection is then performed and up to two RS symbols are corrected in the RS codeword. If errors are detected during CRC or RS processing, the CADUs are annotated accordingly, statistics are updated, and, if error thresholds are exceeded, a message is logged. During grade 3 processing, CADUs identified as fill CADUs are marked in the annotations to be bypassed by the MFPS and a change in the VCID also is detected and marked in the annotations.

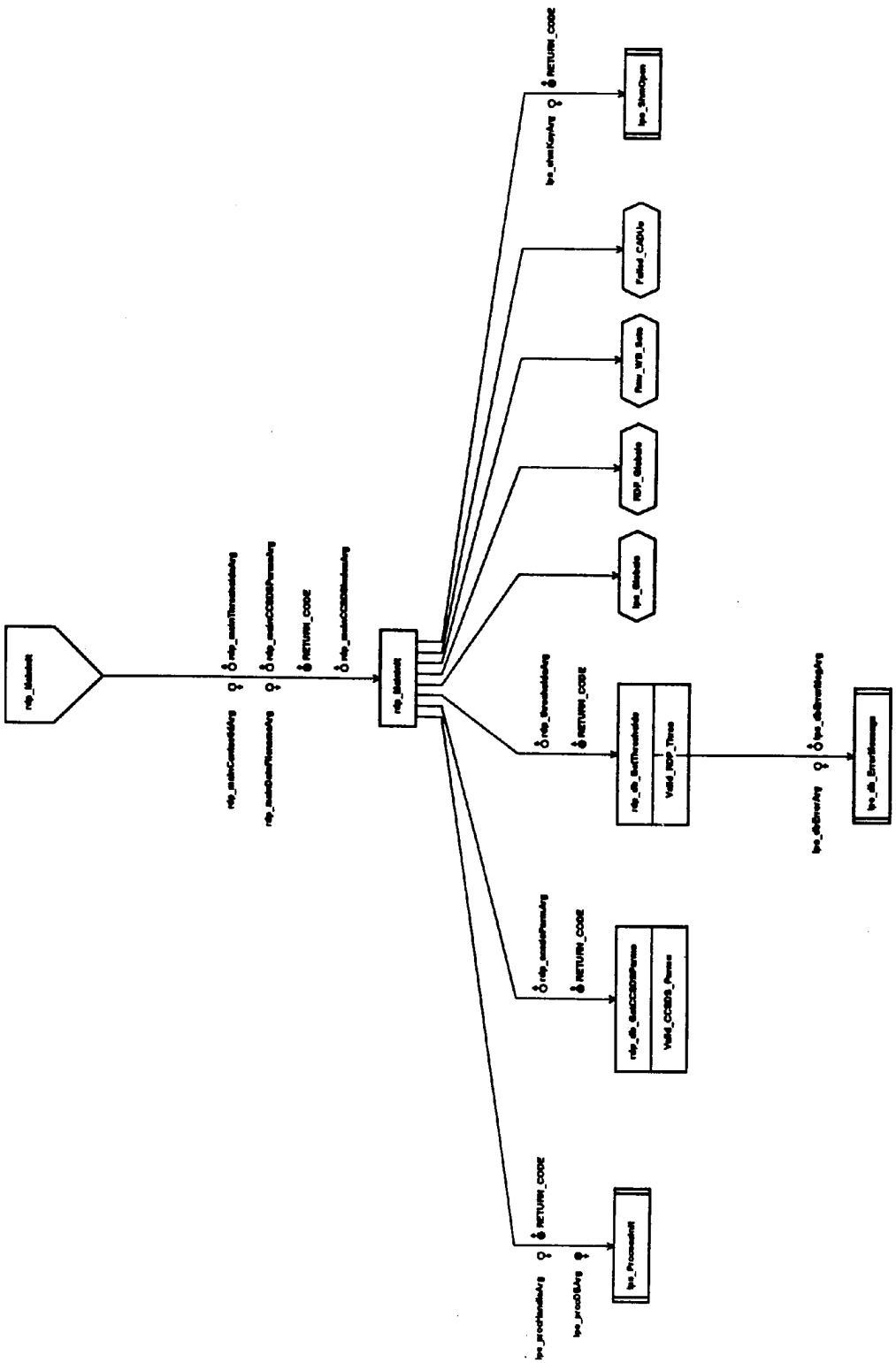


Figure 6–3. rdp_MainInit Structure Chart

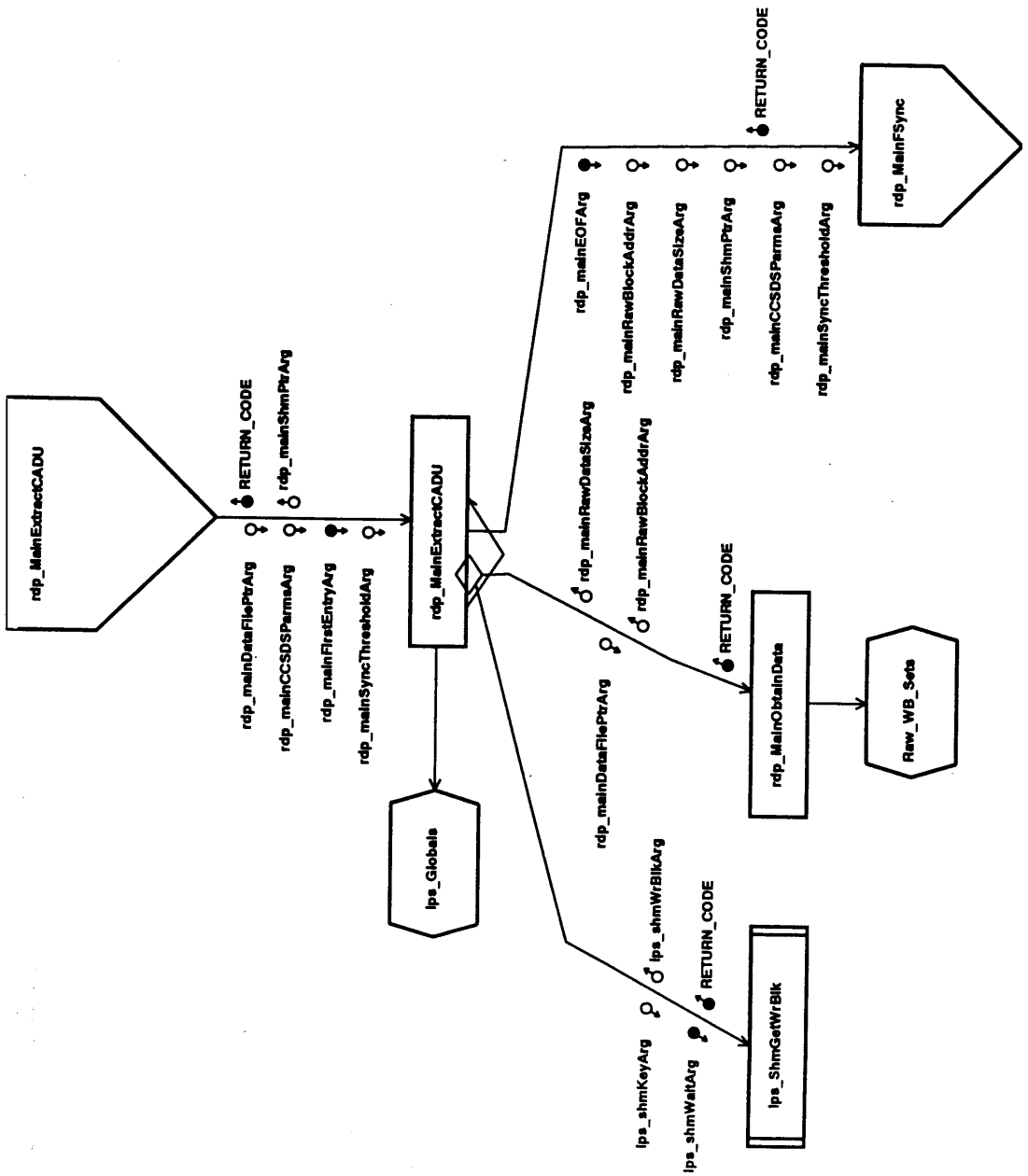


Figure 6–4. rdp_MainExtractCADU Structure Chart

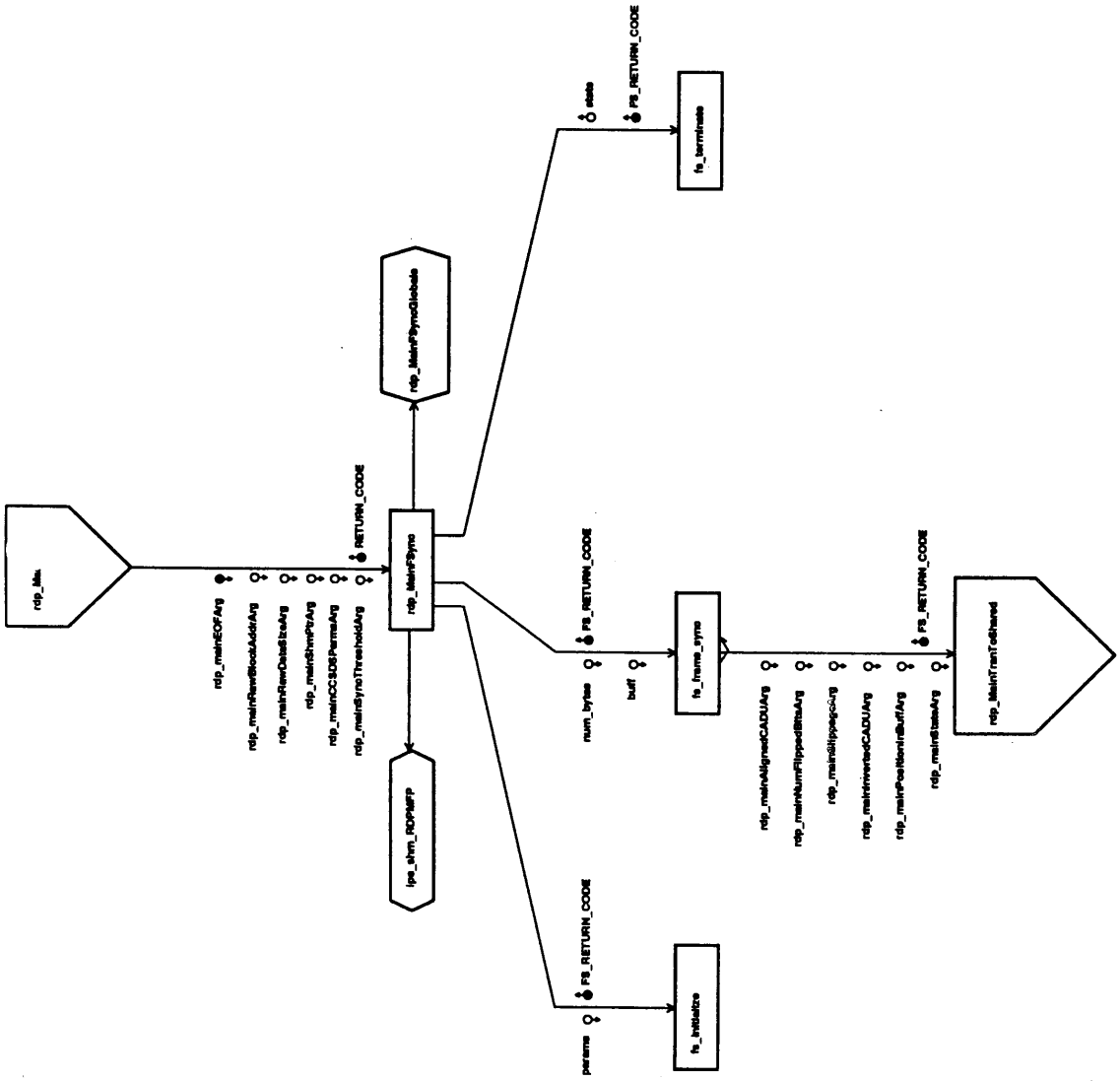


Figure 6–5. rdp_MainFSync Structure Chart

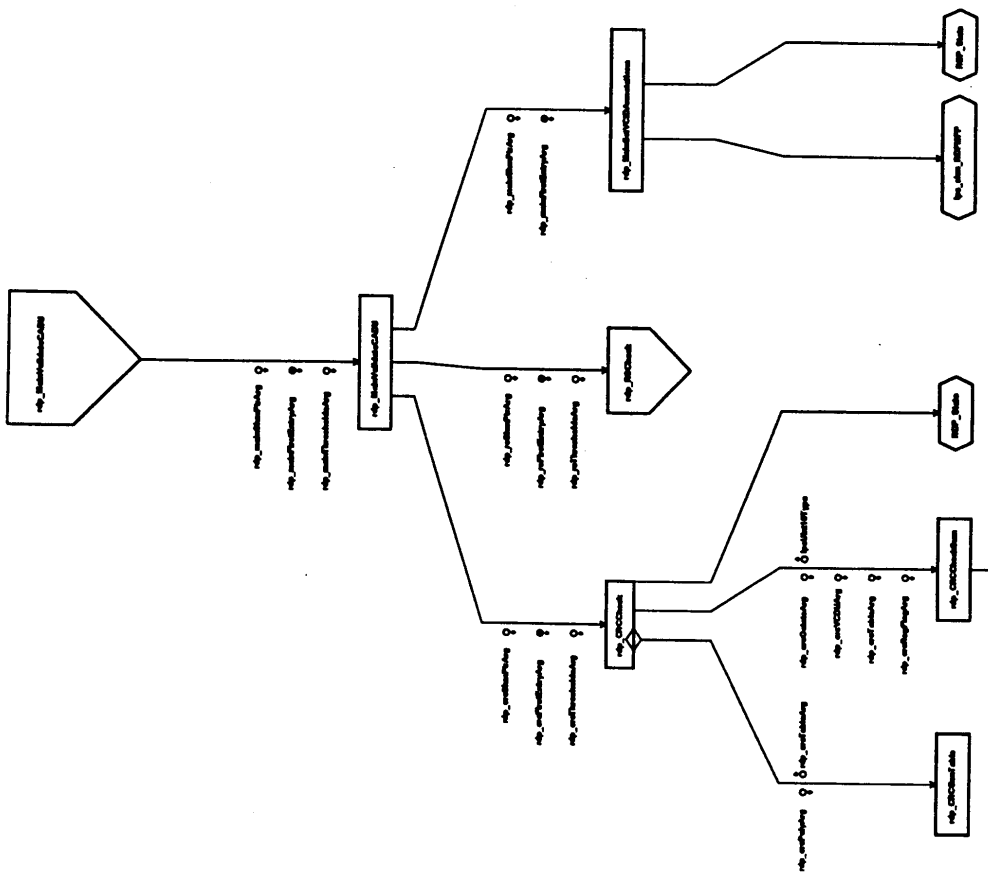


Figure 6–6. rdp_MainValidateCADU Structure Chart

6.3.2.5rdp_BCHDecode

Error detection and correction continues in rdp_BCHDecode (Figure 6–7) only if there are CRC, frame synchronization, RS, or bit slip errors. In addition, BCH error detection and correction will be performed on all CADUs if an appropriate flag is set by the invoking process. Tables needed by the BCH algorithm are either prestored or, in the case of the table needed to solve quadratic equations in Galois field, GF(2^10 – 1), created by rdp_BCHBuildMsnQuadTable. Separate BCH error detection and correction is provided for the mission data zone and the data pointer. The mission data zone and its BCH zone is 8-way interleaved, giving rise to eight BCH code blocks. The pointer area and its BCH zone is just one code block. Thus, there are nine distinct code blocks. rdp_BCHTransposeCADU (Figure 6–8) deinterleaves the mission code blocks and performs error correction and detection on the nine code blocks. The BCH algorithm can correct up to three bit errors per code block for a maximum of up to 27 corrected bits for all nine code blocks. CADUs that have been BCH corrected are identified, as are those that have BCH errors and could not be corrected. The CADUs are annotated to reflect these errors. Statistics regarding BCH error detection and correction are tallied. If the number of BCH uncorrectable CADUs exceeds a predetermined threshold, a log message will be sent.

6.3.2.6rdp_MainGenerateOutput

The output from RDPS processing, which includes the failed CADUs file and the annotated CADUs, is either generated in rdp_MainGenerateOutput (Figure 6–9) or made available to the MFPS. Copies of the annotated CADUs that have failed CRC, RS, or BCH error detection are stored in the failed CADUs file in rdp_MainStoreFailedCADUs. The block is then released to be accessed by the MFPS.

6.3.2.7rdp_db_PutRDPAcctInfo

After all blocks have been processed by the RDPS, rdp_db_PutRDPAcctInfo (Figure 6–10) stores the raw data processing statistics to the database to be used in generating the return-link Q&A report.

6.3.2.8rdp_MainShutdown

When all processing by the RDPS has been completed for a contact period, rdp_MainShutdown (Figure 6–11) is called to perform the cleanup duties. First, the software masks out the termination signals. lps_db_Disconnect is then called to disconnect from the database. The files for the raw wideband sets and the failed CADUs are closed. Finally, the RDPS calls lps_ShmClose to detach from the shared-memory area created for the annotated CADUs.

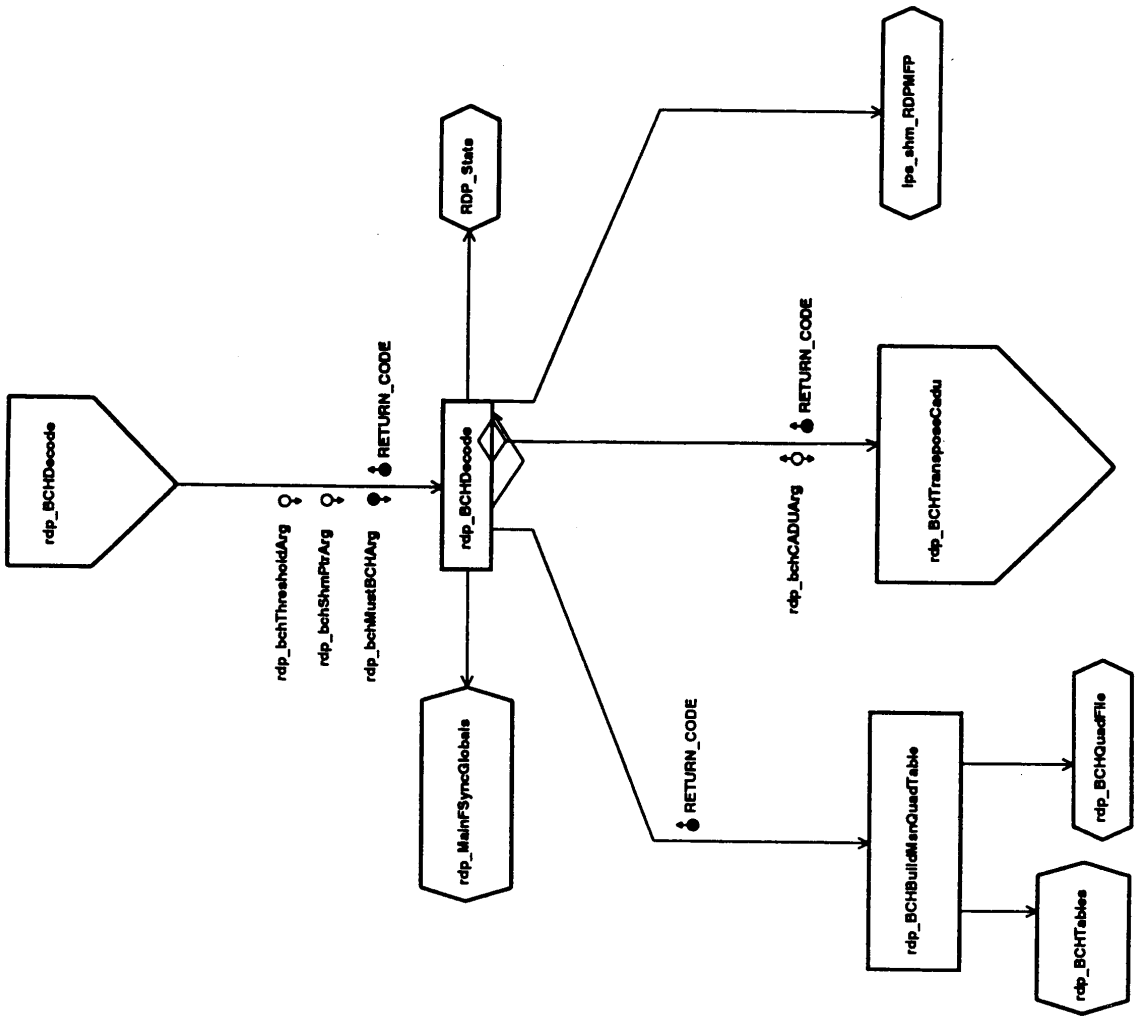


Figure 6–7. rdp_BCHDecode Structure Chart

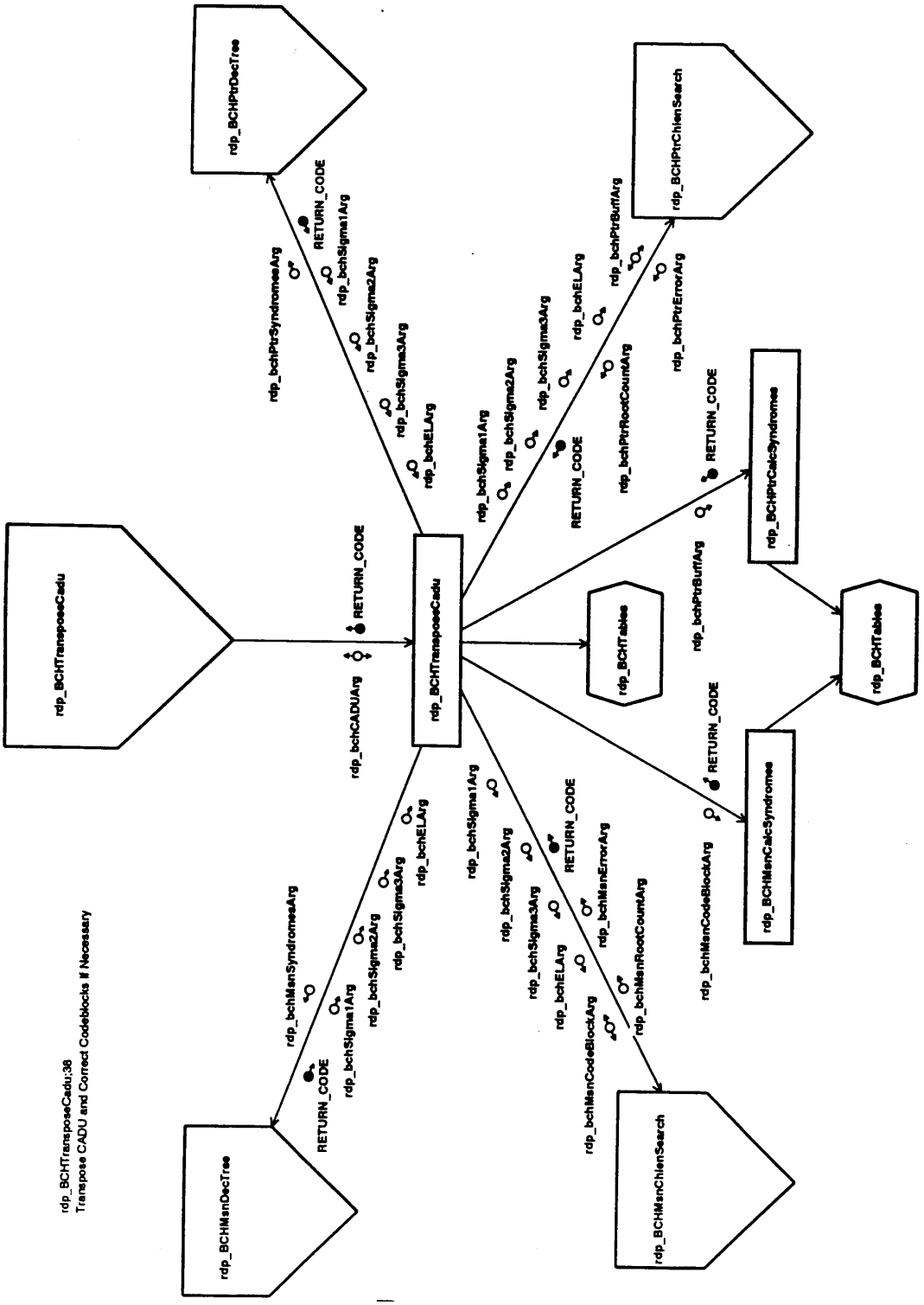


Figure 6–8. rdp_BCHTransposeCADU Structure Chart

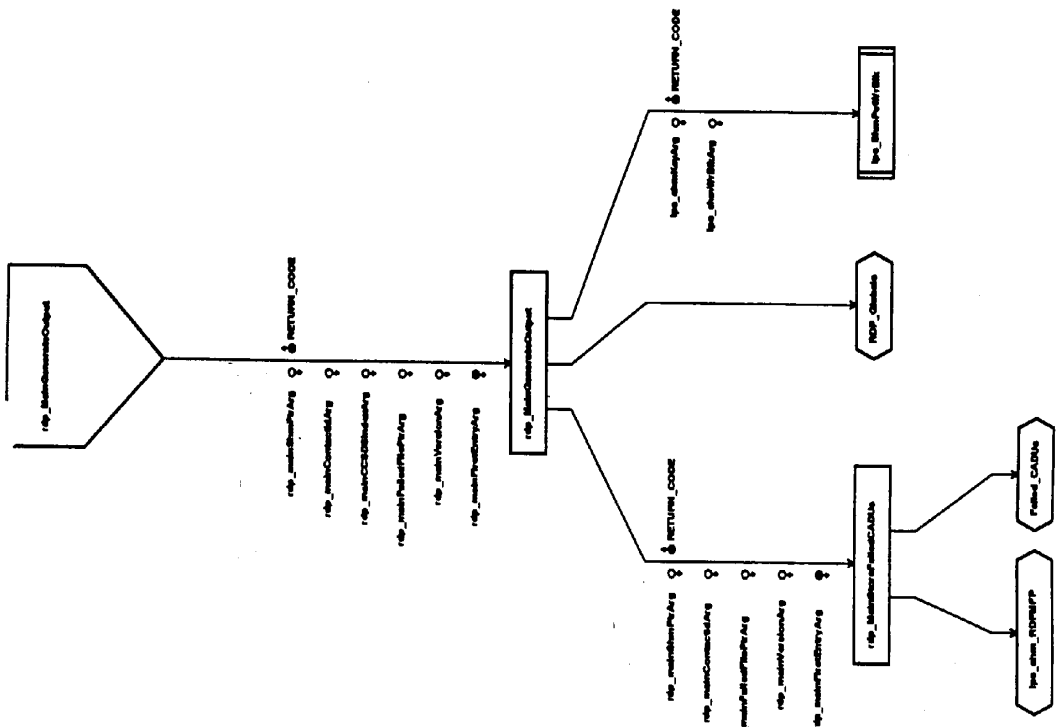


Figure 6–9. rdp_MainGenerateOutput Structure Chart

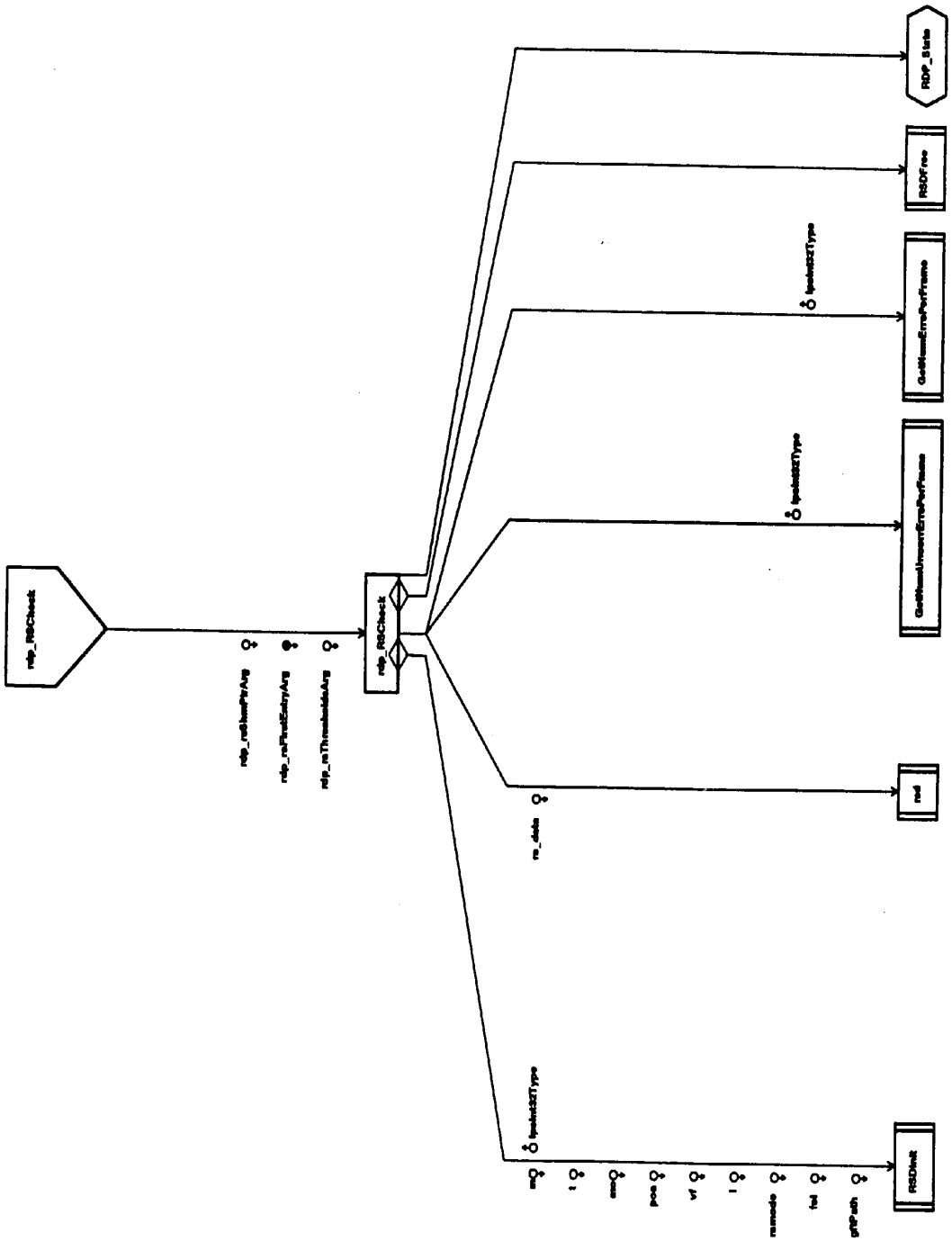
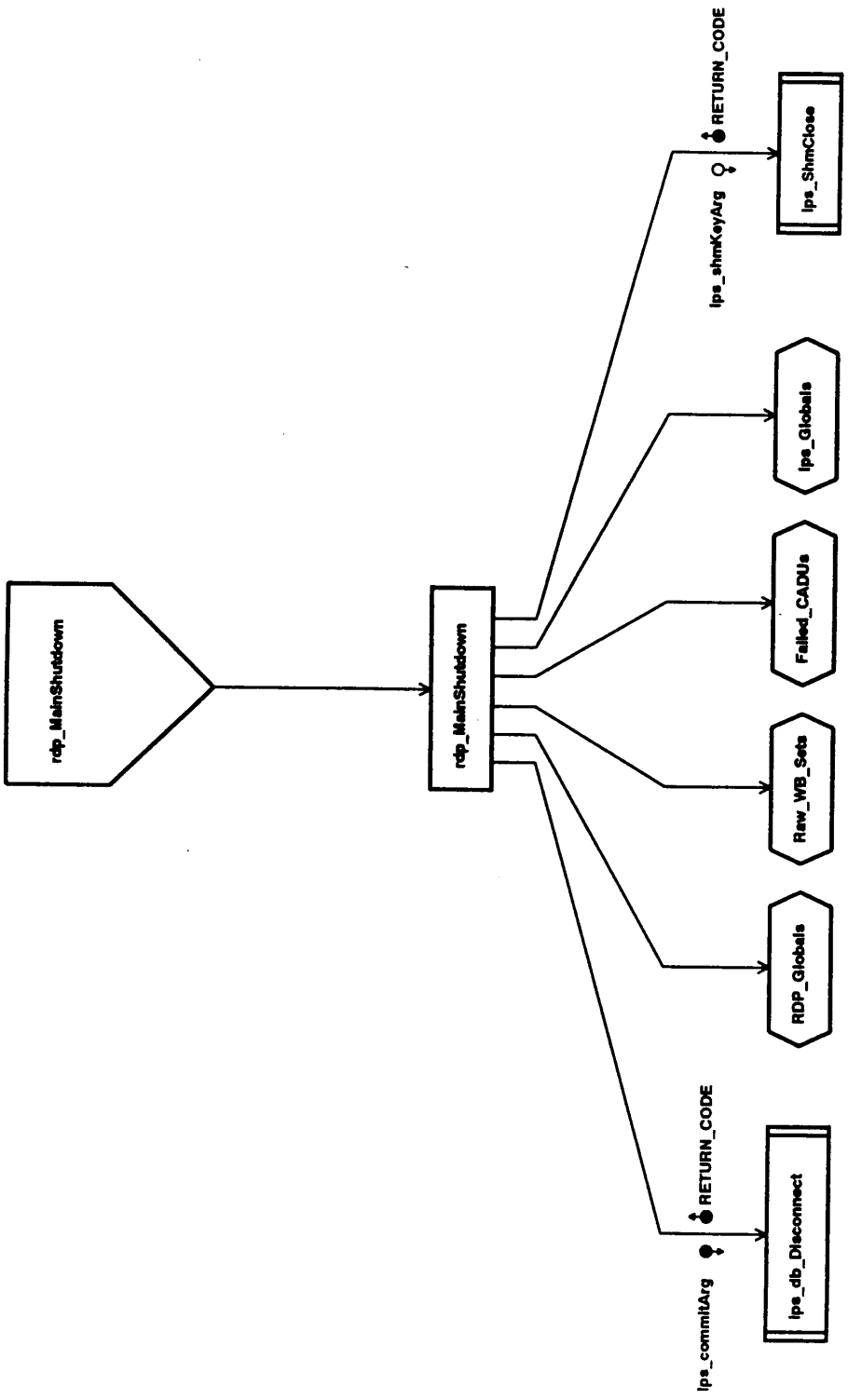


Figure 6–10. rdp_db_PutRDPAcctInfo Structure Chart

Figure 6–11. rdp_MainShutdown Structure Chart


rdp_MainShutdown
rdp_MainShutdown
lps_ShmClose
RETURN_CODE
lps_shmKeyArg
lps_Globals
Failed_CADUs
Raw_WB_Sets
RDP_Globals
lps_db_Disconnect
RETURN_CODE
lps_commitArg

Section 7. Major Frame Processing Subsystem

7.1 Introduction

The MFPS is responsible for identifying a major frame, determining the major frame time and the subinterval, generating aligned band data, and generating the calibration and mirror scan correction data (MSCD) files on a subinterval basis. It also is responsible for extracting the virtual channel data unit (VCDU) status information and passing it to the IDPS. Additionally, PCD bytes are extracted and sent to the PCDS.

The MFPS generates and reports the Level 0R Q&A data and provides status messages to inform the operator of the MFPS processing status.

7.2 Design Overview

This section provides an overview of the MFPS software design. The relationship between the MFPS and other Landsat 7 subsystems is presented, along with a discussion of the assumptions, constraints, and considerations used in the design process.

7.2.1 Subsystem Software Overview

As shown in the MFPS context diagram (Figure 7–1), the MFPS interfaces with the RDPS to receive the annotated CADUs, PCDS to output PCD bytes and related information, and IDPS to output aligned bands and related information.

The MFPS searches the annotated CADUs for an end-of-contact marker, a VCID change, and minor frame counter rollover. These indicators are used as delimiters for collecting a set of CADUs that are expected to contain a major frame. This set of CADUs is then searched at expected locations for the ETM+ major frame synchronization and end-of-line (EOL) codes. If the EOL code cannot be found, no further processing is done on this CADU set. If synchronization and EOL is found, the timecode minor frames are extracted and the major frame time is generated. If the time is not deemed correct, an estimate will be provided. The major frame time is also checked to determine any missing major frames. Any missing major frames are filled with specified fill data. The starting locations of the scene data and the MSCD are output for later processing. The Q&A information for each major frame is entered into the database. The major frame time, VCID, and end of contact are checked to determine whether a subinterval has ended and/or a new subinterval has begun. When a subinterval stop has been detected, subinterval information and Q&A are entered into the database. Q&A is compared against threshold values. The only other database transactions to occur in the MFPS are at MFPS initialization. The database generates a subinterval sequence identification that the MFPS extracts and uses in its processing. Those CADUs that are not placed in a major frame are saved in a trouble file.

7.2.2 Design Considerations

This section presents the design drivers relevant to the MFPS software design. It describes the assumptions, software reuse strategy, and required operational support that influence the design of the MFPS software.

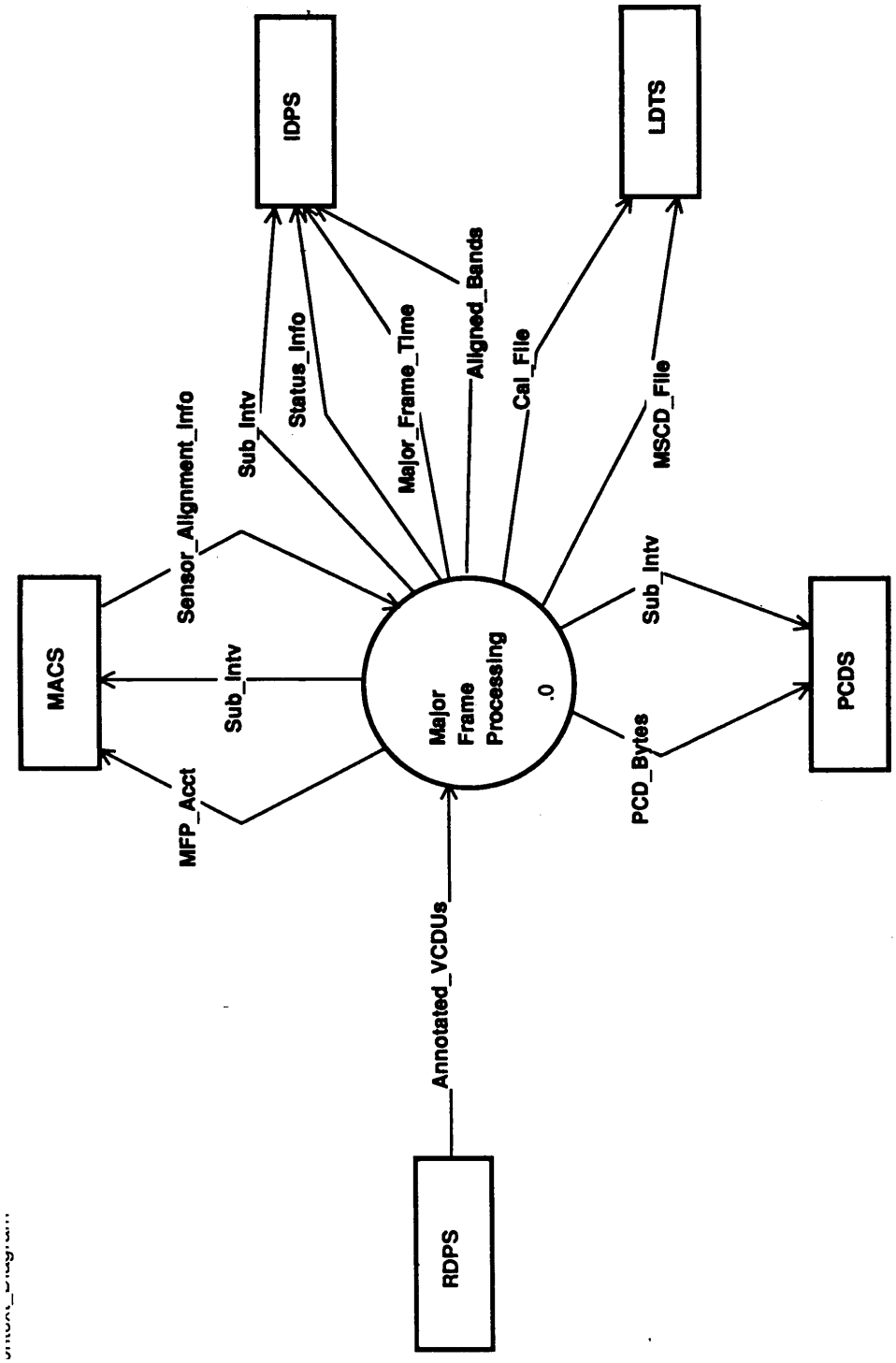


Figure 7–1. Major Frame Processing Context Diagram

7.2.2.1 Assumptions

At this point in the design phase, assumptions have been made that impact the software design and remain in effect until notified otherwise.

- The subinterval identifier generated from the database utility will never be zero.
- The MFPS expects the CADUs it received from the RDPS to be ordered.
- The MFPS expects the spacecraft clock to be accurate. The 40-to-1 bit encoding and the BCH error checking reduce the probability of a bad transmission. Therefore, the subinterval break, when it occurs on the basis of major frame, is assumed to be accurate.
- The number of bytes for one band and detector combination is assumed to be constant and equal to the maximum alignment value plus the average number of bytes expected from the data in one major frame, as specified in the data format control book (DFCB).

7.2.2.2 Open Issues

No issues remain open at this time.

7.2.2.3 Operational Support

This section presents components of the software design that support the normal and contingency operations of the MFPS.

7.2.2.3.1 Start Up the MFPS Software

The MFPS will be “forked” as a child process by the MACS. During initialization, the MFPS will

- Set a signal trap to catch any signals that would cause MFPS to terminate abnormally.
- Connect to the database and remain connected throughout the contact period.
- Access the database during initialization and at the declaration of the beginning and end of a subinterval.
- Connect to IPC mechanisms that communicate with the IDPS, PCDS, and RDPS.

If the MFPS is unable to obtain any of the services requested during initialization, with the exception of accessing the thresholds from the database, a message will be logged and a return code sent to mfp_Main. The MFPS will halt processing.

7.2.2.3.2 Abnormal Termination

The MFPS sets a signal trap to catch any signals that may cause the MFPS to terminate abnormally. If a signal is trapped, a signal handler is executed to clean up the temporary files and IPC mechanisms.

7.2.2.3.3 Create Level 0R Output Files on a Subinterval Basis

The MFPS creates the calibration and MSCD files on a subinterval basis. This operational scenario is supported by the mfp_L0RFilesGen module (described in Section 7.3.2.4).

7.2.2.3.4 Support Reprocessing of Data

The MFPS does not differentiate between data captured from the satellite and placed on disk and data being input directly from tape. Therefore, reprocessing will be handled in the same manner as processing.

7.2.2.4 Software Reuse Strategy

This section identifies external components that may be reused by the MFPS, as well as common components of the MFPS software that may be useful to other Landsat 7 subsystems. Table 7–1 describes component types, Table 7–2 describes ease of use classifications, and Table 7–3 defines type and ease of use for each reusable component.

7.2.3 Subsystem Error Handling

The errors and processing exceptions that may be encountered by the MFPS are discussed in the LPS Users Guide.

Table 7–1. Component Type

Component Type	Description
Design	The algorithm only may be reused.
Design/Code	The algorithm and the code may be reused.
New Development	The unit has not yet been developed.

Table 7–2. Ease of Use

Ease of Use	Description
Major modifications required	Substantial modification is required to reuse the component.
Minor modifications required	Minor modification is required to reuse the component.
No modifications	No modification is required to reuse the component.

Table 7–3. Reusable Components

Reusable Component	Type	Ease of Use
mfp_AlignBands	Design/Code	No modifications
mfp_ChckSplitMnf	Design/Code	No modifications
mfp_CondenseDataGrp	Design/Code	No modifications
mfp_Deint	Design/Code	No modifications
mfp_FillMostBands	Design/Code	No modifications

mfp_FillBand6	Design/Code	No modifications

7.3 Subsystem Design

7.3.1 Top-Level Model

The top-level model of MFPS is shown in Figure 7–2. mfp_Main is the main routine for the MFPS. From the MACS it receives the file version number and the contact sequence identification. mfp_Main calls mfp_MainInit to handle memory requirements; database access; IPC connections to the RDPS, PCDS, and IDPS; and signal handler installation and to initialize variables for the start of processing. If any of these fail, the MFPS will halt. mfp_MainValidateMjf is responsible for collecting CADUs, identifying the major frame, generating the major frame time, and compiling Q&A statistics for the CADUs on a major frame basis. mfp_MainDetermineSub checks each major frame to place it within a subinterval. mfp_MainPcdStatusProc is responsible for formatting the necessary information and sending it to the PCDS. It also processes the remaining status information and stores it in the global area. The calibration and MSCD files are generated in mfp_MainL0RFilesGen. The aligned band data is generated and made available to the IDPS in mfp_MainBandGen. mfp_MainQ&ASubGen collects all Q&A statistics on a subinterval basis and adds Q&A statistics to the database on a major frame and subinterval basis. Statistics exceeding the threshold value trigger a routine to send an error message to the message log file. mfp_MainCleanup is activated at the close of MFPS processing, both normal and abnormal. mfp_MainCleanup disconnects from the database, detaches and deallocates all memory, closes IPC mechanisms, and closes any open files.

7.3.2 Detailed Module Design

7.3.2.1 mfp_MainInit

mfp_MainInit (Figure 7–3) allocates all local memory. It connects to the shared memory between MFPS and RDPS, MFPS and PCDS, and MFPS and IDPS. The MFPS connects to the database and queries it for parameters and threshold values. Q&A variables are set to zero. A prefilled data area is created for use in filling missing major frames. A pseudo CADU is created for use when the MFPS needs to substitute missing CADUs in the data stream. If the MFPS is attempting to initialize while raw data capture is occurring, it will terminate.

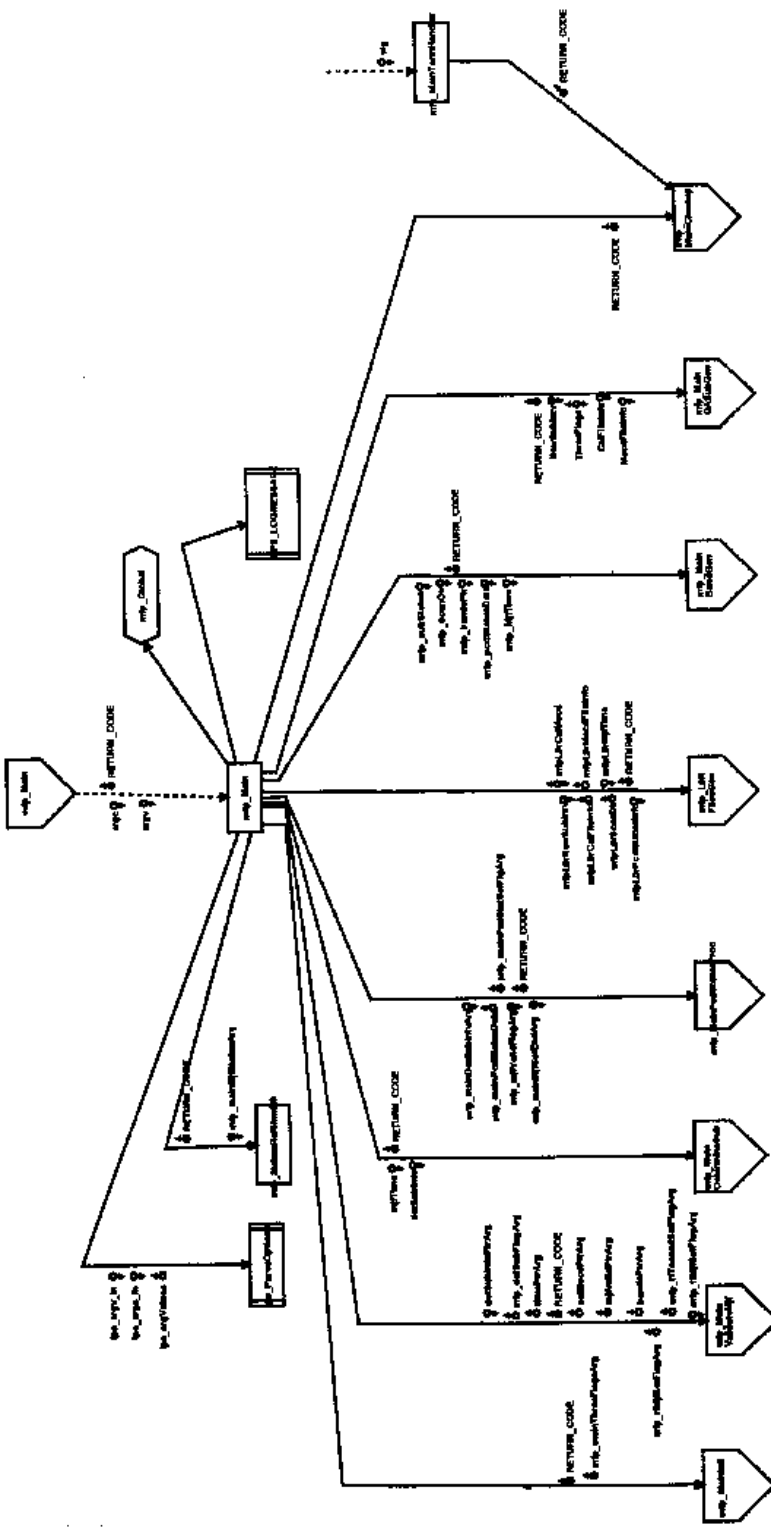
7.3.2.2 mfp_MainValidateMjf

mfp_MainValidateMjf (Figure 7–4) is responsible for collecting CADUs for a major frame, validating the major frame, generating the major frame time, and compiling Q&A statistics for the CADUs on a major frame basis. If timecode minor frames cannot be used to generate the major frame time, the module generates an estimated time. This module also saves the CADUs that do not belong to a major frame in a separate area of memory for a trouble file.

mfp_MainIdentifyMjfSet (Figure 7–5) is responsible for identifying the CADUs for a major frame. It starts by calling mfp_MainFindMjfStart, and then calls mfp_MainAdd2Set.

mfp_MainFindMjfStart (Figure 7–6) is responsible for checking the CADUs in the shared memory to find a pair of CADUs with identical VCIDs and a drop in minor frame counter values so the process of collecting CADUs for a major frame can start. This module saves CADUs that do

Figure 7–2. mfp_Main Structure Chart



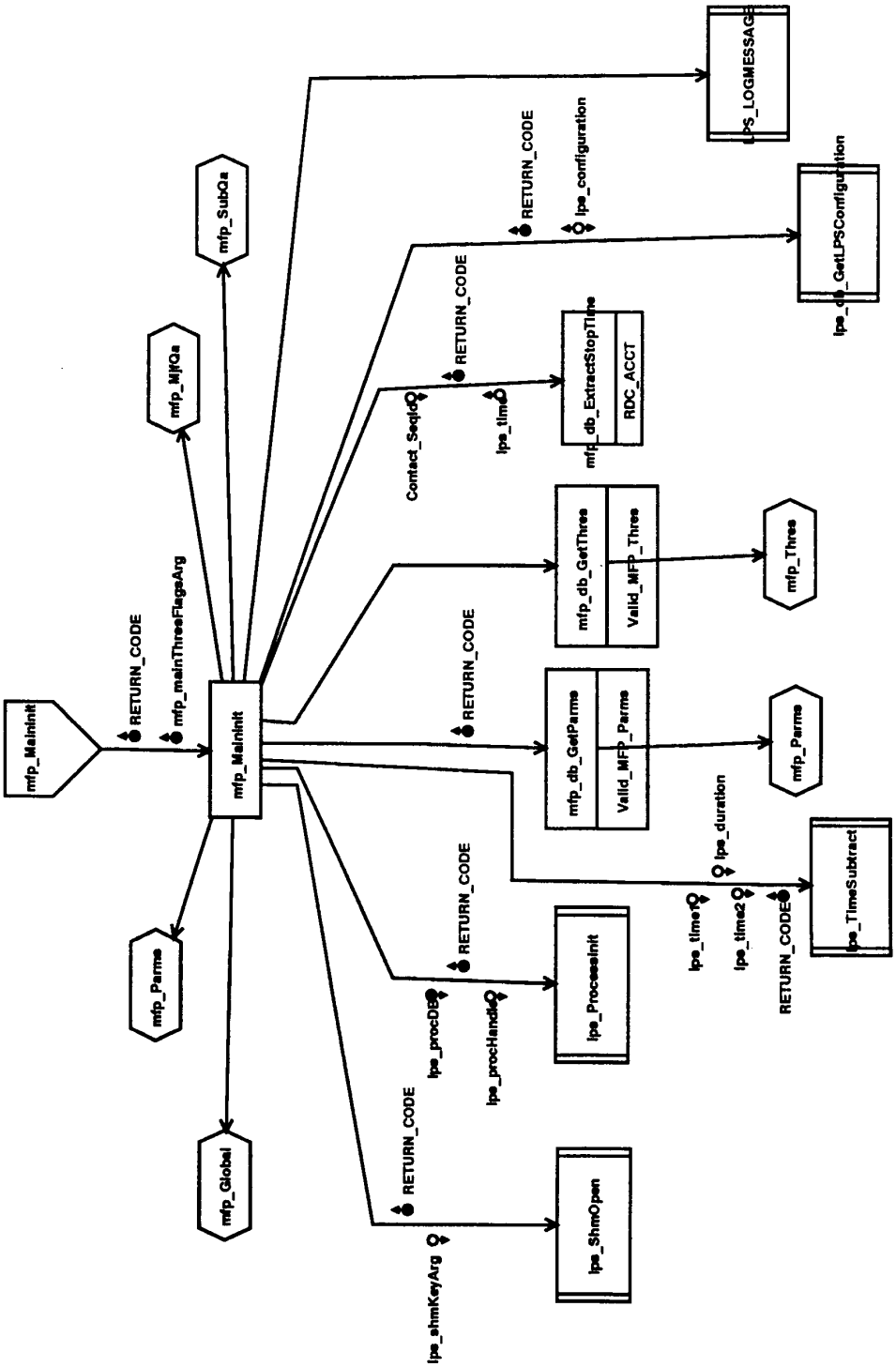


Figure 7–3. mfp_MainInit Structure Chart

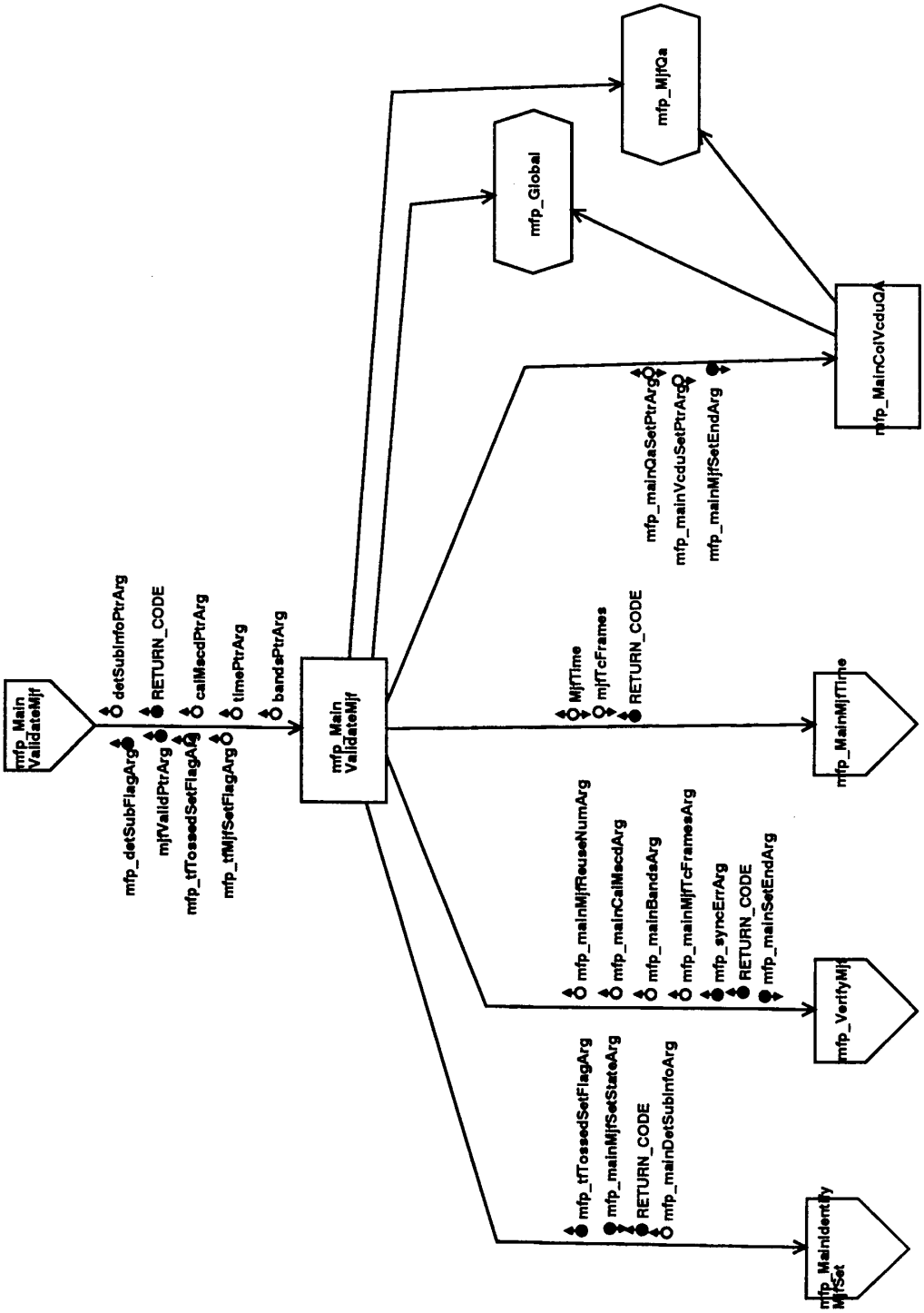


Figure 7–4. mfp_MainValidateMjf Structure Chart

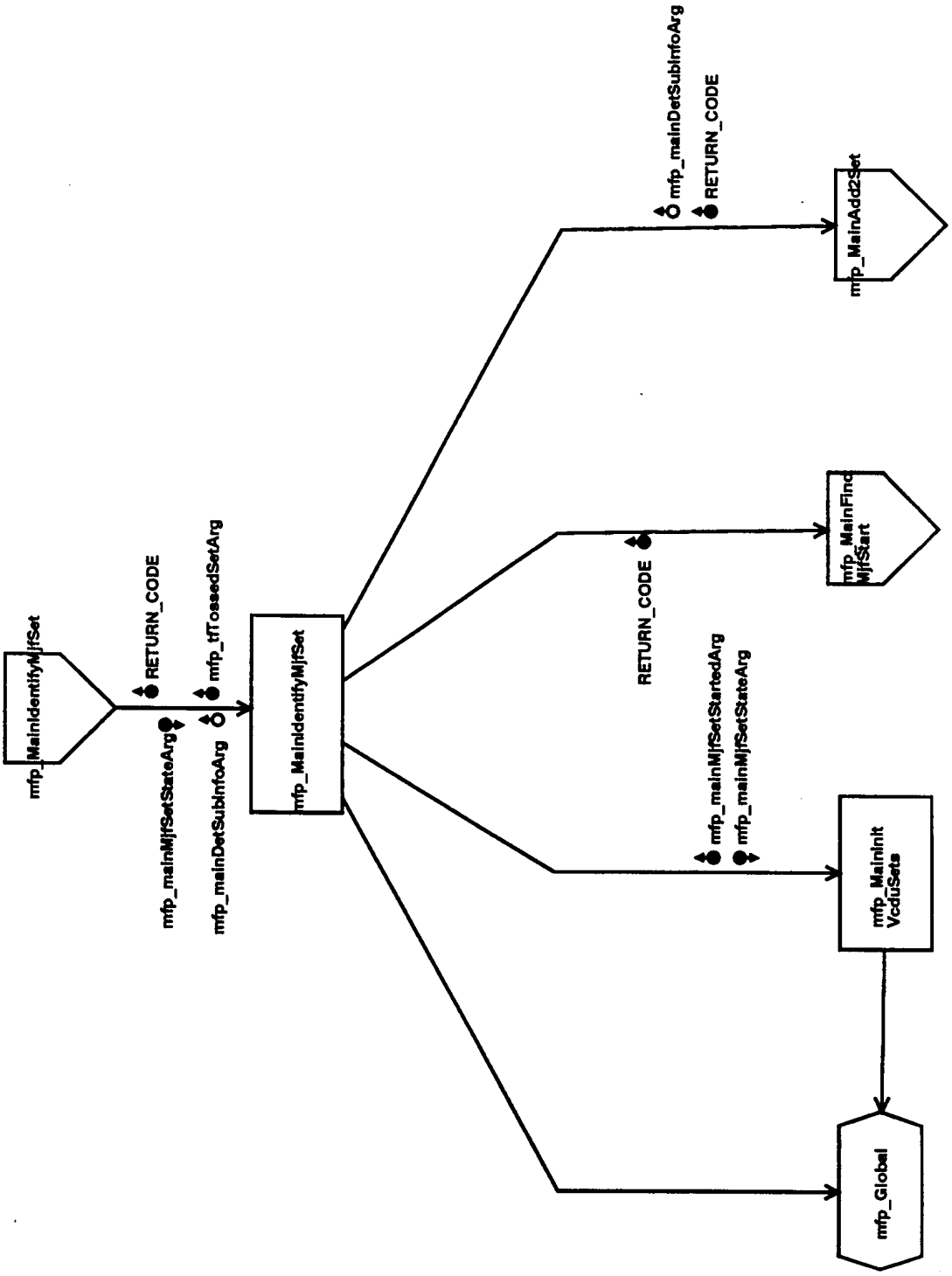


Figure 7–5. mfp_MainIdentifyMjfSet Structure Chart

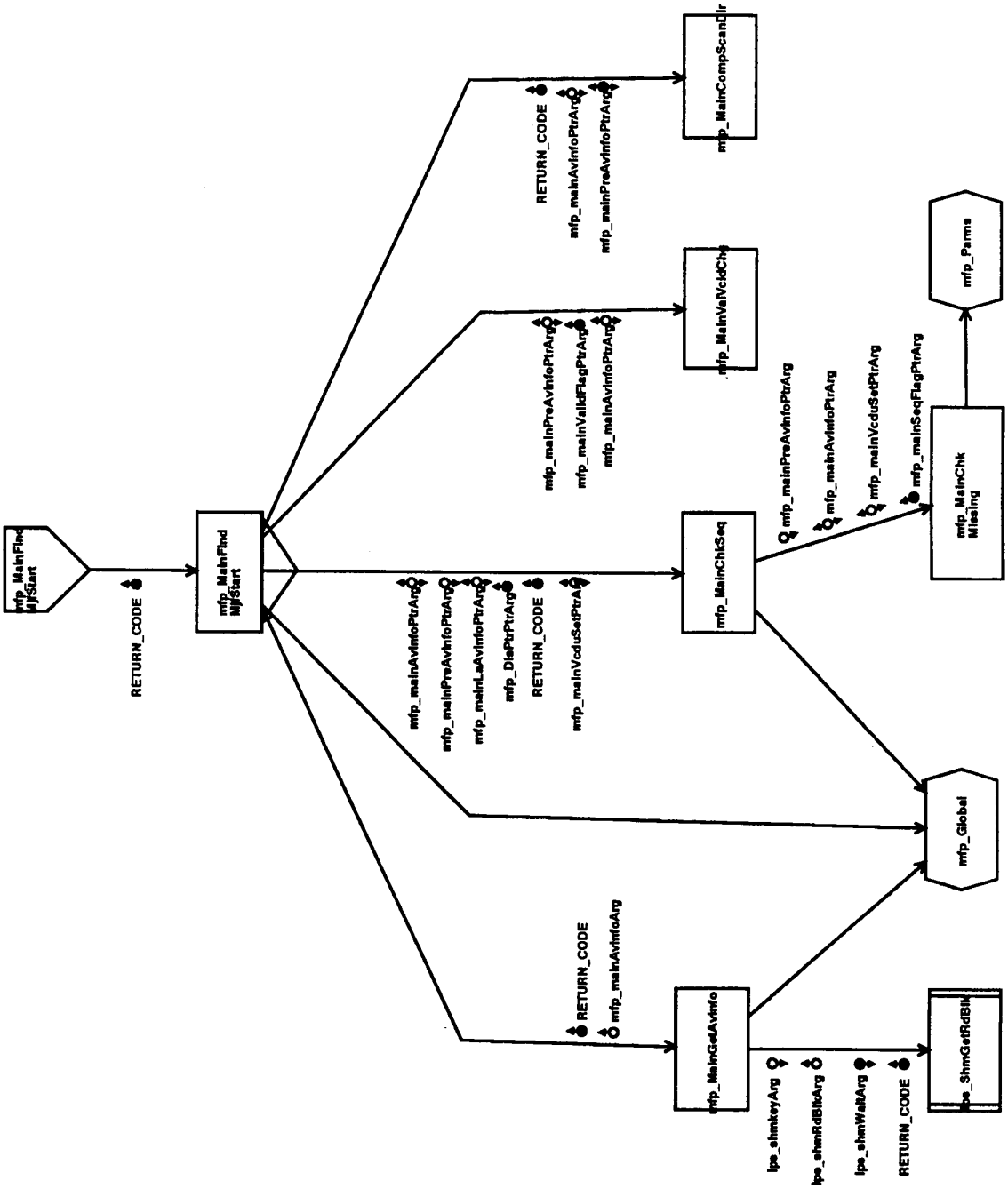


Figure 7–6. mfp_MainFindMjfStart Structure Chart

not belong to a major frame in a separate CADU set. The process of checking CADUs stops when end of contact (EOC) is detected, the CADU set is full, or the desired pair of CADUs for starting the Major Frame set is found. This module also generates data about missing CADUs.

mfp_MainAdd2Set (Figure 7–7) is responsible for collecting the CADUs for a major frame once the initial set of CADUs for the major frame has been identified. The process of collecting CADUs for the major frame stops when end of contact is detected, the VCID changes, minor frame counter rollover occurs, or the number of CADUs in the major frame exceeds a prespecified threshold value without locating a minor frame counter rollover. This module also substitutes the missing CADUs with predefined pseudo CADUs for ease of deinterleaving bands. In addition, the module generates data about missing CADUs.

mfp_MainMjfTime (Figure 7–8) is responsible for extracting the major frame time from six timecode minor frames. If the major frame time is found to be invalid, the time is estimated using the previous major frame time and the major frame period. The routine ensures that the start of a contact period uses a valid major frame time, meaning the major frame time is actual and not estimated.

7.3.2.3 mfp_MainDetermineSub

mfp_MainDetermineSub (Figure 7–9) determines the subinterval start and stop times using one of three criteria: the current major frame time, a VCID change, or the end of contact indicator. To declare a new subinterval, the VCID must have changed or the current major frame time must exceed the previous major frame time by a specified time interval, which is a parameter that can change at the start of each contact period. The end-of-contact indicator only ends an existing subinterval.

At the start of a subinterval, the MFPS sends the subinterval information to the database. When the end of a subinterval is declared, the MFPS updates the database to include the stop time for that subinterval.

7.3.2.4 mfp_L0RFilesGen

mfp_L0RFilesGen (Figure 7–10) calls the mfp_MscdExtract and the mfp_CalExtract routines to extract the MSCD and calibration data on a major frame basis for the given subinterval. Each file is opened and closed on a subinterval basis. Error handling procedures are used for unsuccessful file opens, file closes, or data appends to their corresponding files.

mfp_MscdL0rExtract (Figure 7–11) extracts the MSCD from two contiguous minor frames immediately following the EOL code. This routine extracts the second-half scan error (SHS_Err), which consists of 12 groups of five data words each; the first-half scan error (FHS_Err), which consists of 12 groups of 5 data words each; and the scan direction, which consists of eight groups of five data words each. mfp_CondenseDataGrp is used to condense each group (40 bits) to a single bit. If either SHS_Err or FHS_Err are found invalid, a bit flag will be set indicating its status. If the scan direction is found invalid, it is estimated from the previous scan direction. The appropriate header information, MSCD data, and major frame time is written to the MSCD file on

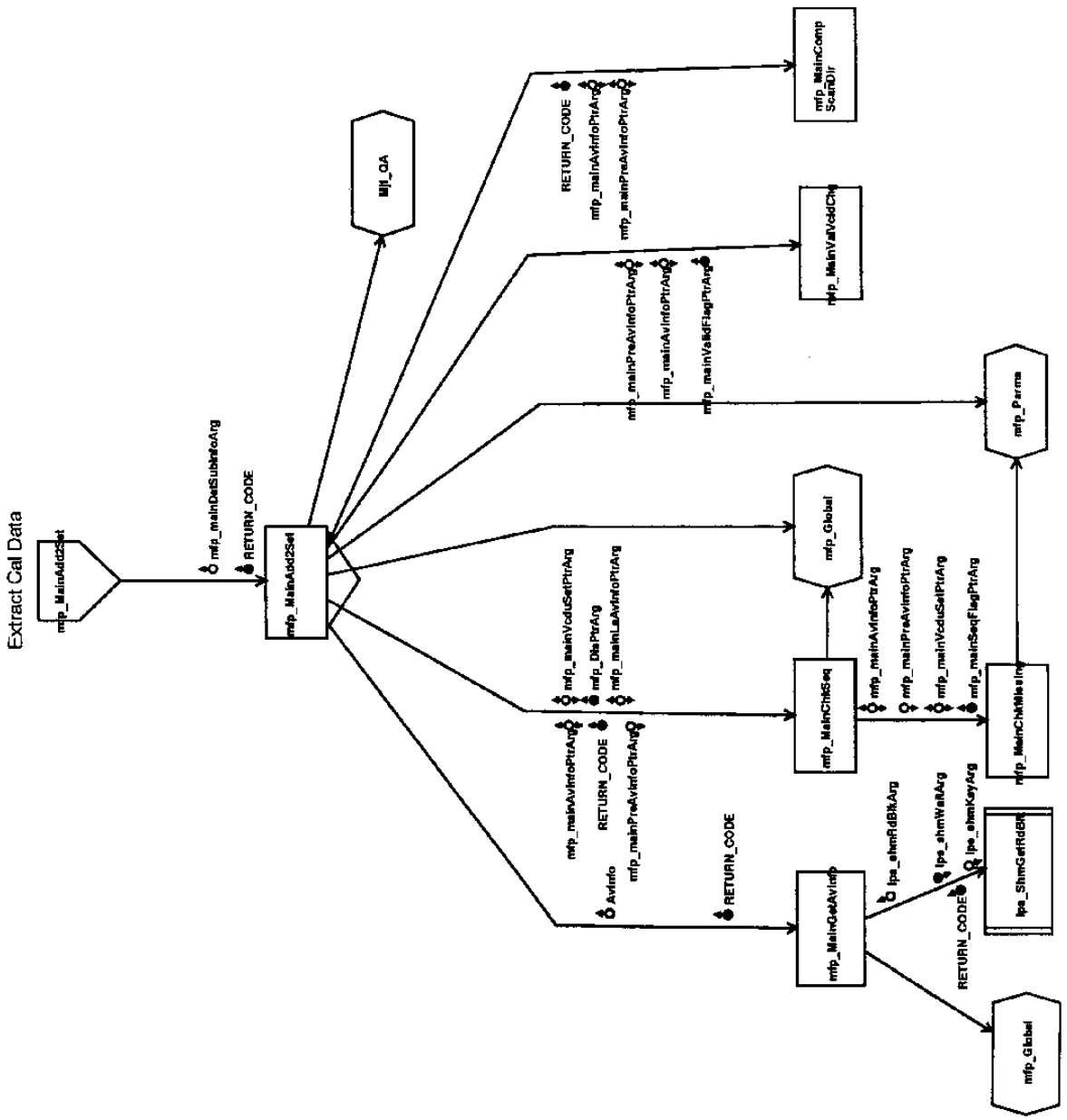


Figure 7–7. mfp_MainAdd2Set Structure Chart

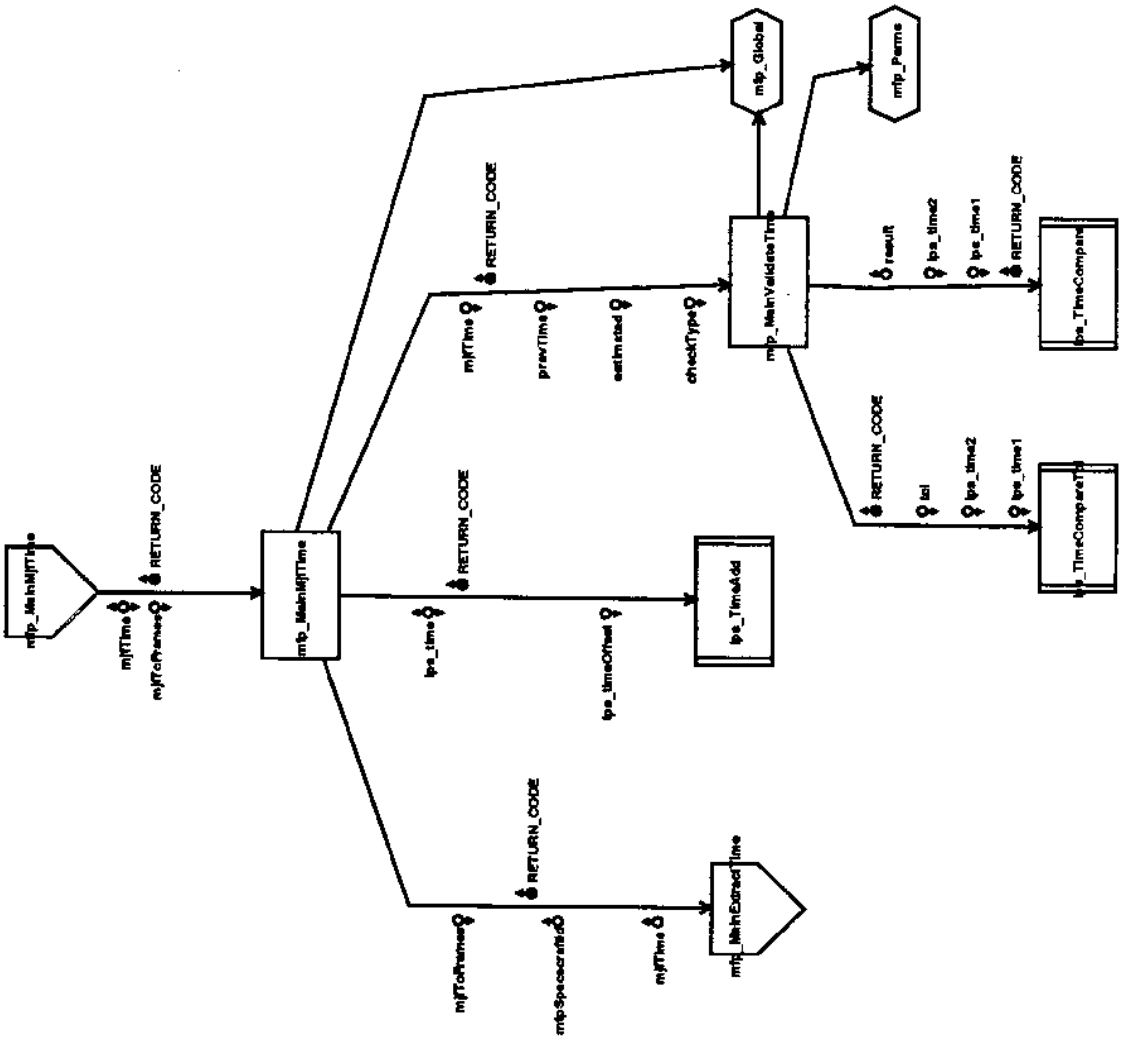


Figure 7–8. mfp_MainMjfTime Structure Chart

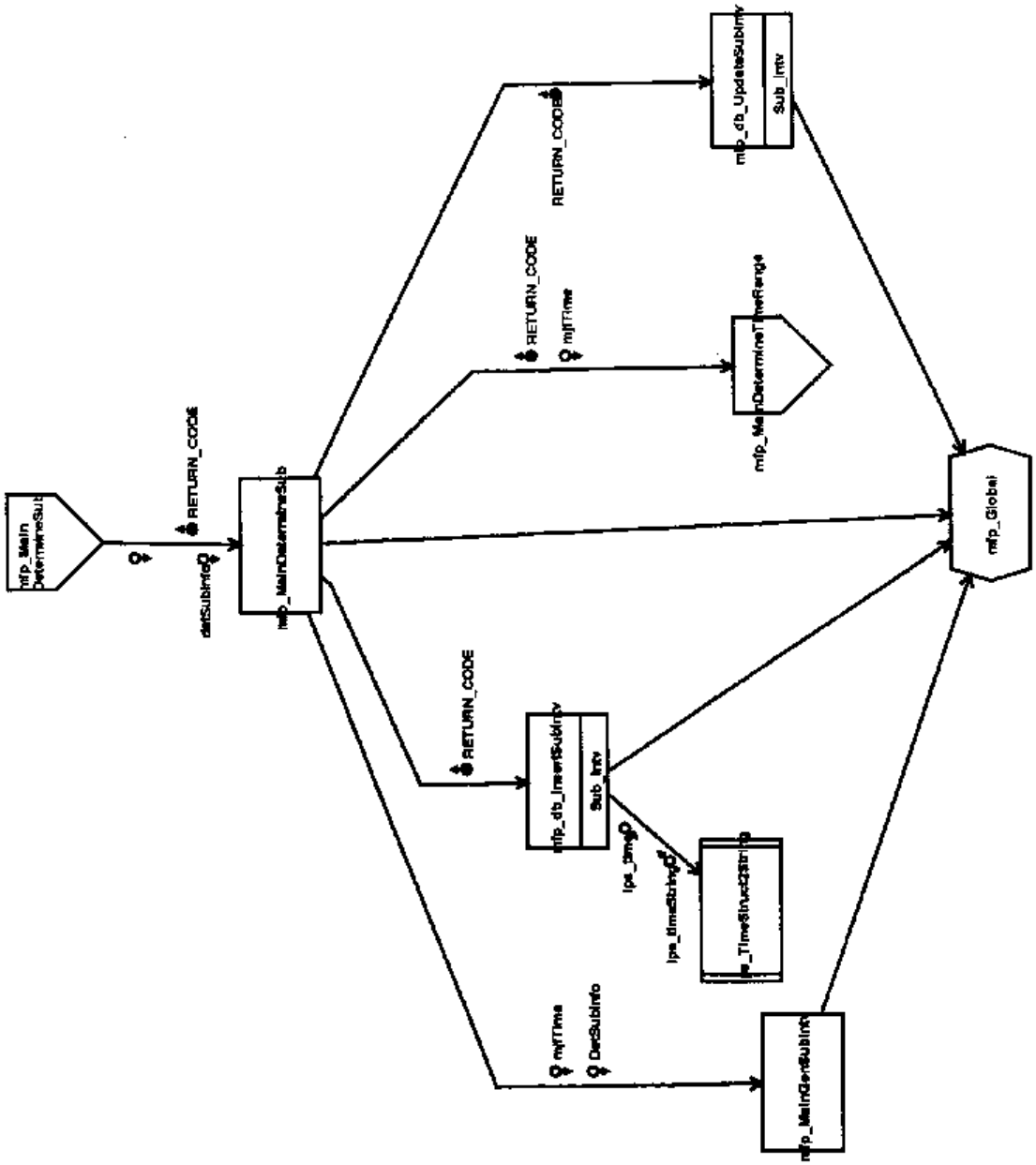


Figure 7–9. mfp_MainDetermineSub Structure Chart

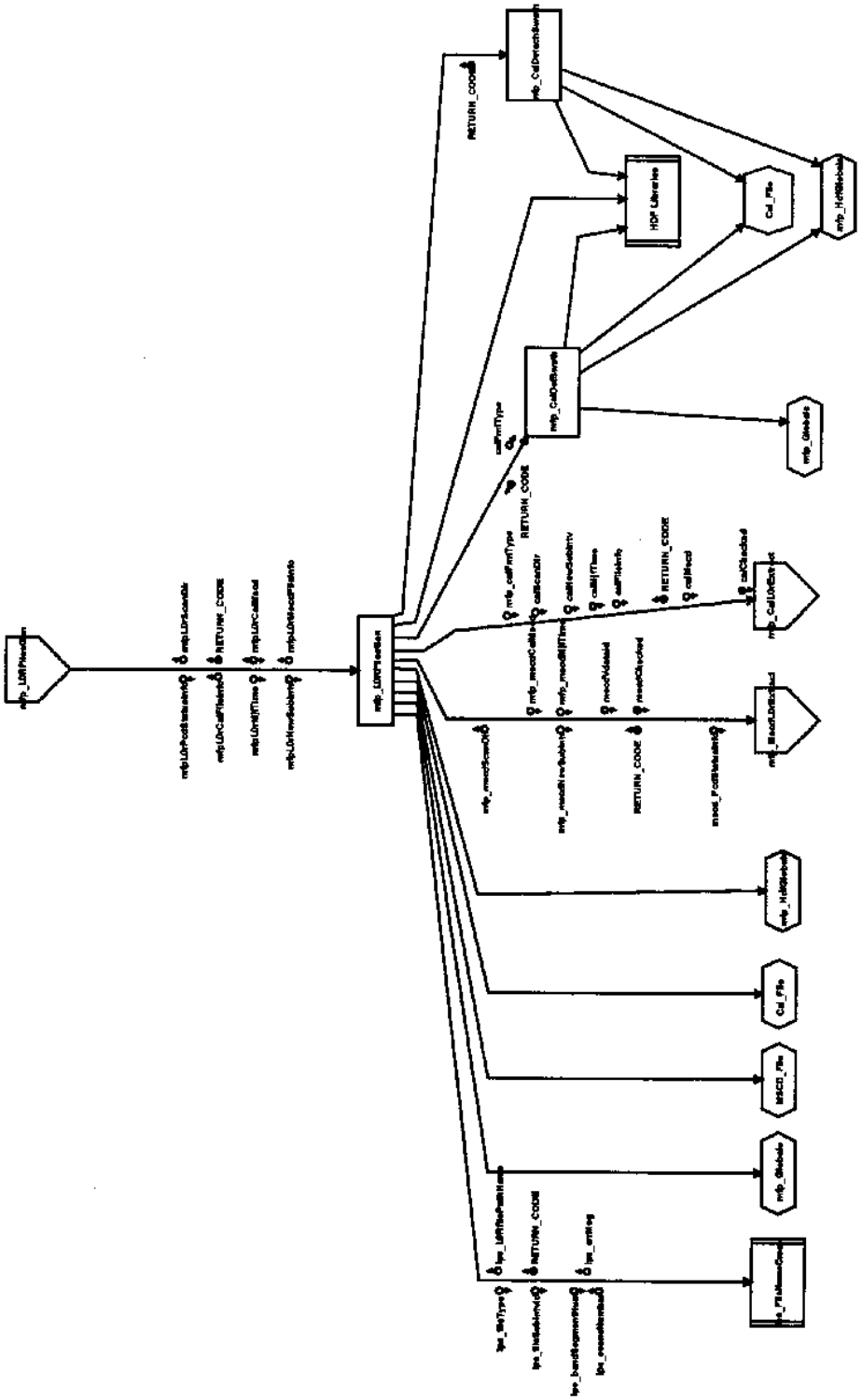


Figure 7–10. mfp_L0RFilesGen Structure Chart

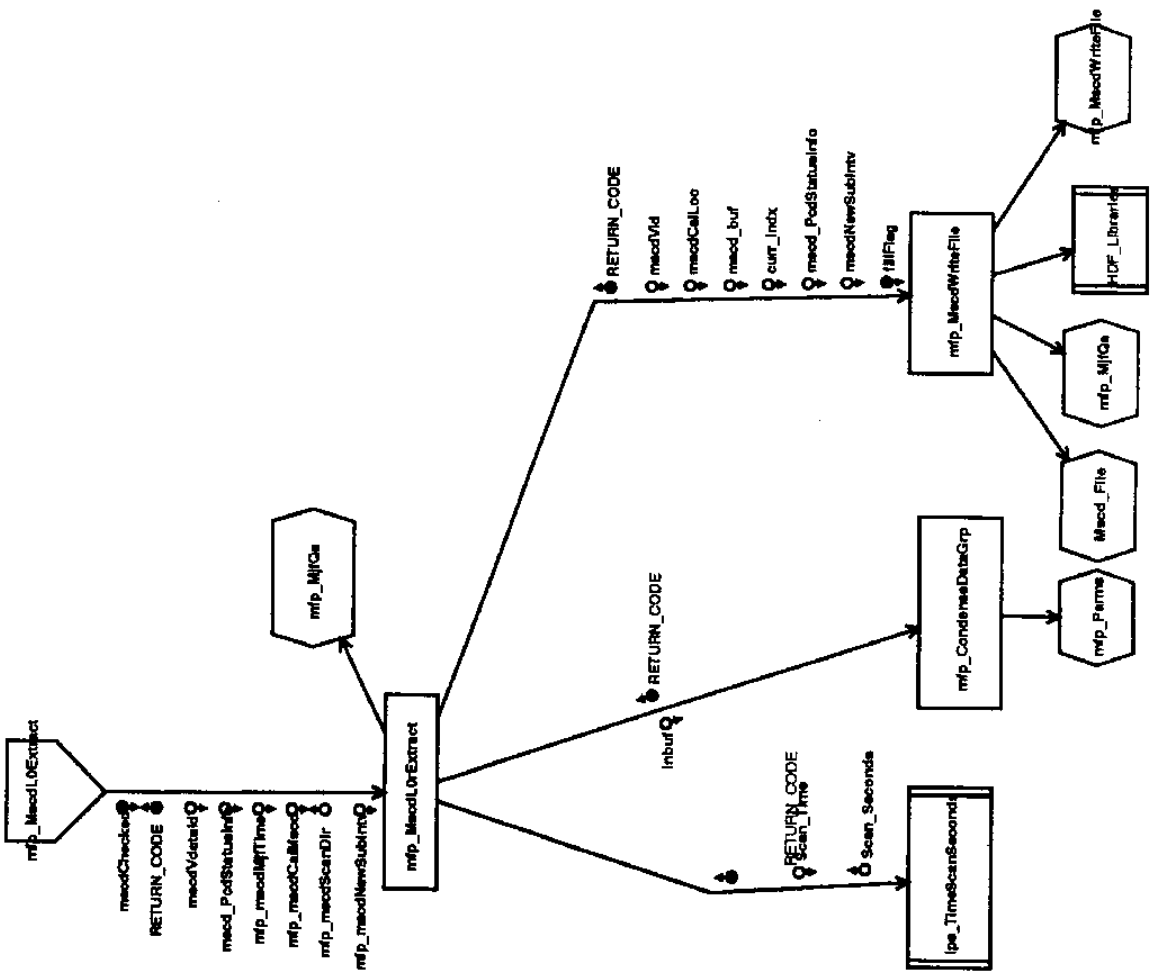


Figure 7–11. mfp_MscdL0rExtract Structure Chart

a major frame basis. The format of the MSCD file follows the file format provided in the LPS output file DFCB.

mfp_CalExtract (Figure 7–12) extracts, deinterleaves, and aligns the calibration data. The calibration data immediately follows the MSCD. This routine extracts the calibration data and calls routines defined in the mfp_mainBandsGen to deinterleave and align the calibration data. Split minor frames, minor frames that span two CADUs, and partial minor frames are handled accordingly by the deinterleave and align routines. The appropriate header information, calibration data, major frame time, and status information for each major frame is written to the calibration file on a major frame basis. The format of the calibration file follows the file format provided in the LPS output file DFCB.

7.3.2.5 mfp_MainBandGen

mfp_MainBandGen (Figure 7–13) generates the aligned band data and transmits it to the IDPS. If a major frame is declared missing, the entire shared memory for that major frame is filled with the predefined fill value and made available to the IDPS. Missing major frames are handled prior to extracting scene data from the current major frame. After the missing major frames are processed, the scene data can be extracted from the current major frame. The alignment information is processed first. The fill value is set at the beginning and end of the shared-memory area for each aligned band and detector set according to the sensor alignment information. Band 6 data is processed first and extracted from all minor frames within the major frame set. The remaining scene data is extracted according to format 1 or format 2 for each minor frame between the timecode and the EOL.

7.3.2.6 mfp_MainQASubGen

mfp_MainQASubGen (Figure 7–14) accumulates the Q&A information on a subinterval basis. The subinterval Q&A is entered into the database and then reinitialized to zero for the next subinterval. The current major frame Q&A is added to the subinterval Q&A and the current major frame Q&A is reinitialized to zero for the next major frame.

Some subinterval statistics are compared to predetermined threshold values. If any of these exceed the threshold value, an error message is sent to the message log file.

At the close of a subinterval, the MSCD filename and the calibration data filename are stored in the database for Q&A purposes.

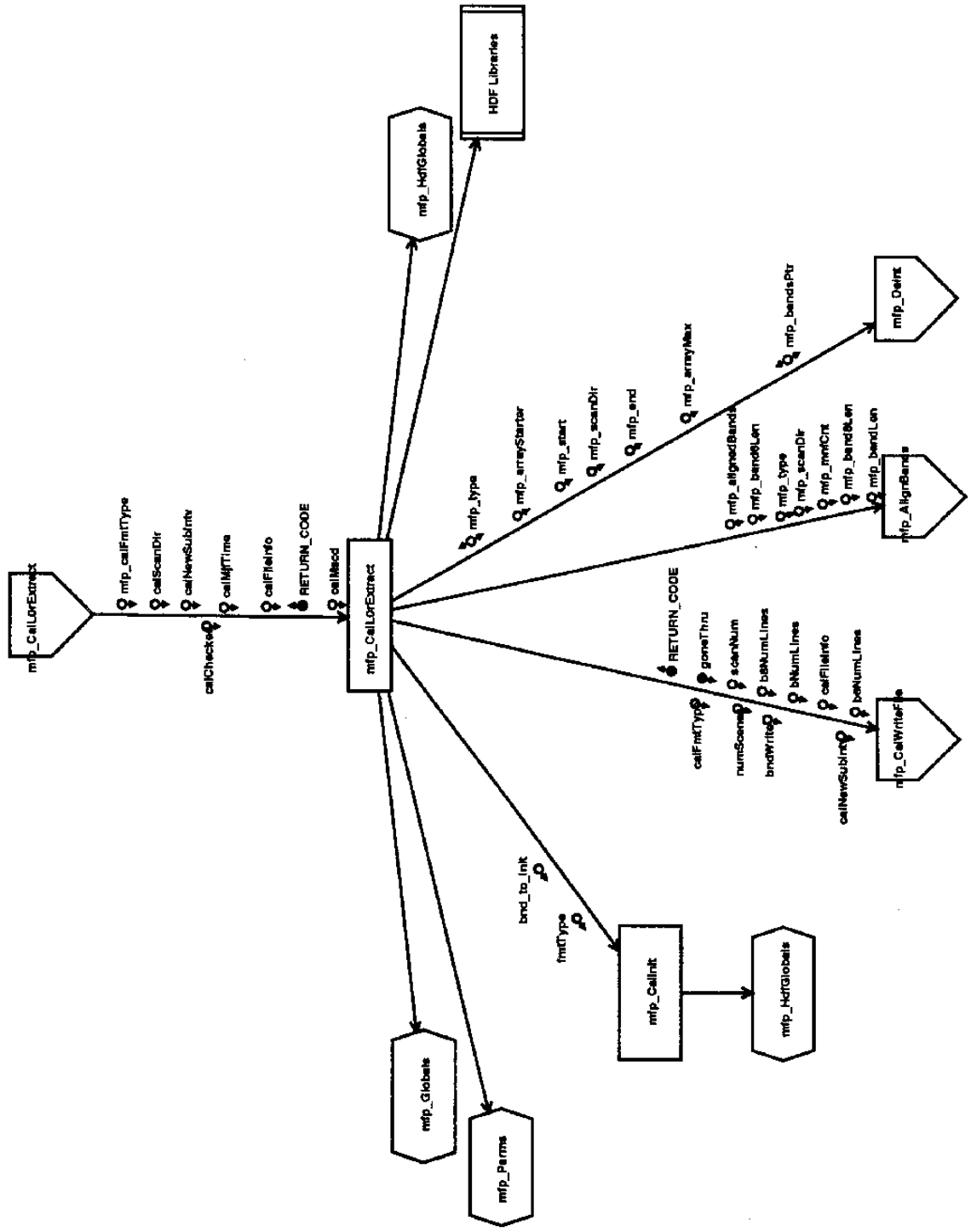


Figure 7–12. mfp_CalExtract Structure Chart

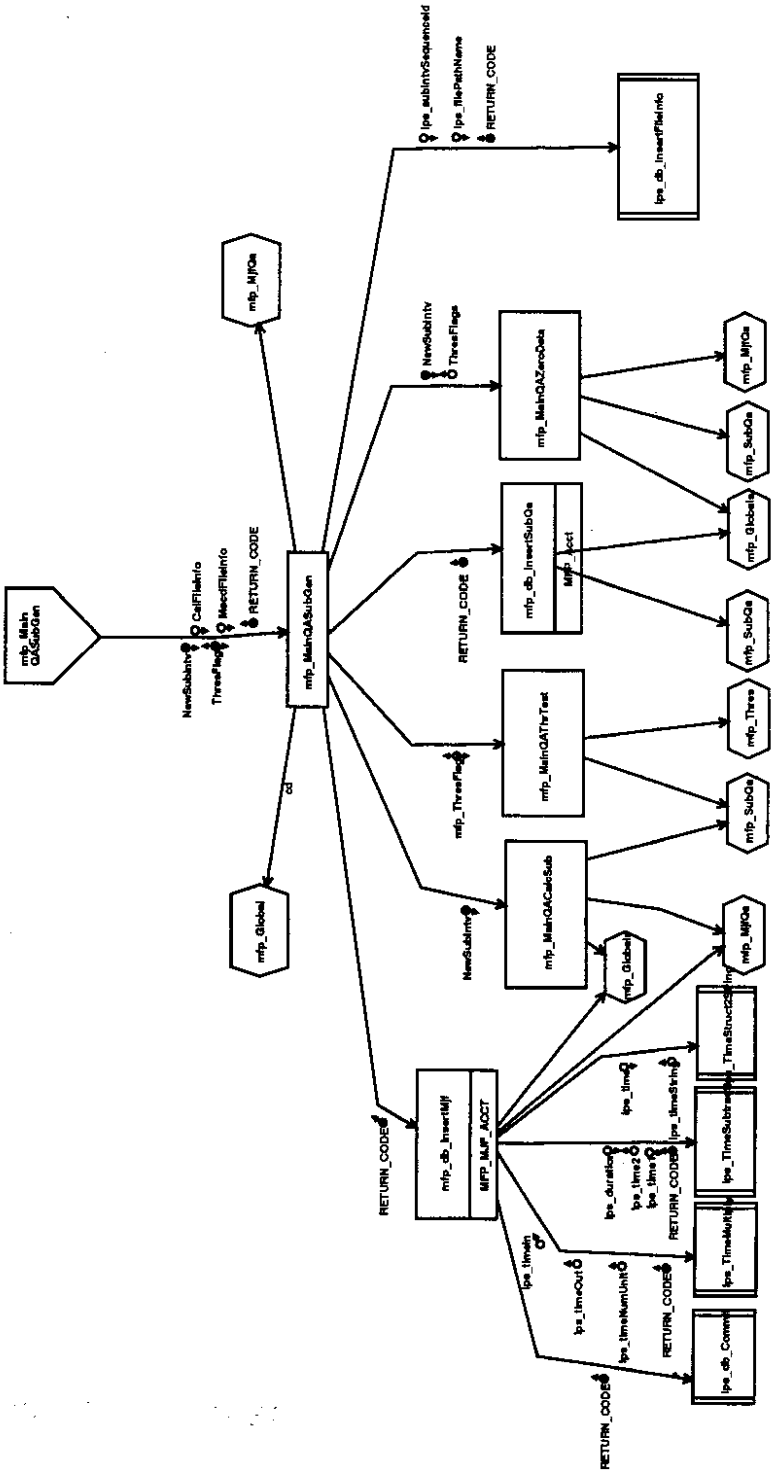


Figure 7–13. mfp_MainBandGen Structure Chart

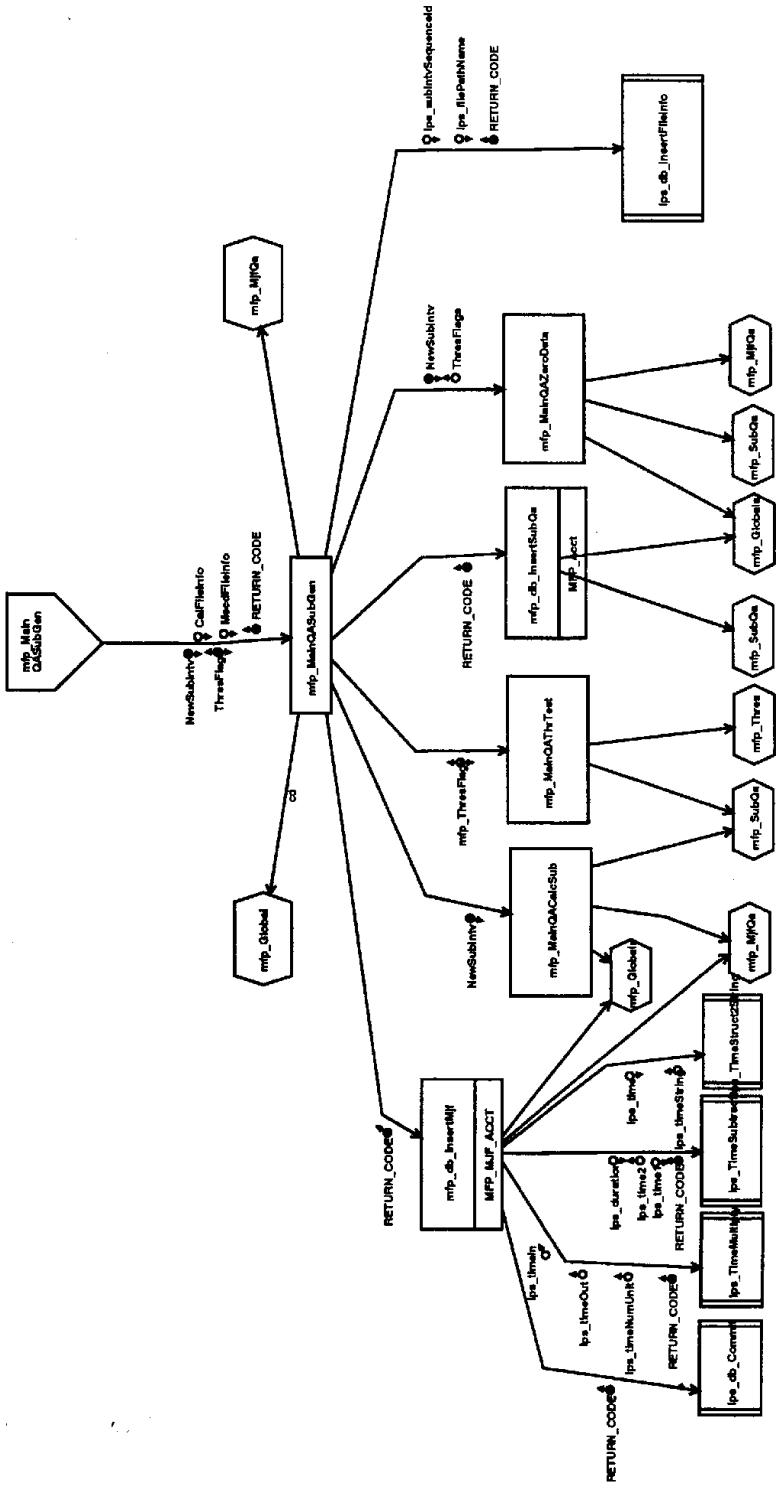


Figure 7–14. mfp_MainQASubGen Structure Chart

Section 8. Payload Correction Data Processing Subsystem

8.1 Introduction

The PCDS is responsible for building PCD cycles and using them to generate PCD files on a subinterval basis. The PCD cycles contain mission-related telemetry that can be used to geometrically correct ETM+ imagery. The PCDS also is responsible for identifying each ETM+ scene in accordance with the Worldwide Reference System (WRS) scheme. Sun azimuth and elevation, horizontal display shift, and calibration door activity status are determined and reported for each identified scene center. Scene information is sent to the IDPS and the MACS for further processing. The PCDS stores Q&A information in the LPS database and provides status messages to inform the analyst of the PCDS processing status.

8.2 Design Overview

This section provides an overview of the PCDS software design. The relationship between the PCDS and other Landsat 7 subsystems is presented, along with a discussion of the assumptions, constraints, and considerations used in the design process.

8.2.1 Subsystem Software Overview

As shown in the PCDS context diagram (Figure 8–1), the PCDS interfaces with the MFPS to receive PCD information (PCD_Info). PCD_Info contains a subinterval identification (Sub_Intv_Id) and an end of contact period flag (End_Of_Contact_Flag). In addition, approximately 642 sets of four PCD words (PCD_Bytes) and the count of any missing VCDUs (Num_Missing_VCDU) from the ETM+ major frame are included.

The PCDS extracts the PCD data word, in triplicate, from PCD_Bytes. A majority vote is performed to determine the actual PCD data word. The PCDS uses the data word to build PCD minor frames. If any of the data words are not available due to missing VCDUs, a fill value is used in place of the missing data word. Counts of the minor frames that contain fill values and the majority vote failures are maintained on a subinterval basis. If either count exceeds operator-selected thresholds, a status message (PCD_Assemble_Cycle_Status) is sent to the MACS to inform the operator of the event. The PCDS thresholds are stored in the LPS database by the MACS for retrieval by the PCDS.

The PCD minor frames are used to build PCD major frames. Any missing PCD minor frames are replaced by a fill value. The PCD major frames are used to build PCD cycles. If any PCD major frames are missing from the PCD cycle, a fill value is used as a replacement. Q&A generated during the building of PCD minor frames, PCD major frames, and PCD cycles is reported to the MACS as a part of the PCD accounting information (PCD_Acct) on a subinterval basis.

The PCDS uses the data from the PCD cycles and the scene parameters (Valid_Scene_Parms) that are stored in the database by the MACS to identify WRS scenes. The scene identification (Scene_Info) is sent to the IDPS. Scene identification and additional scene parameters (Scene_Parms) are stored in the LPS database as a part of PCD_Scene_Acct.

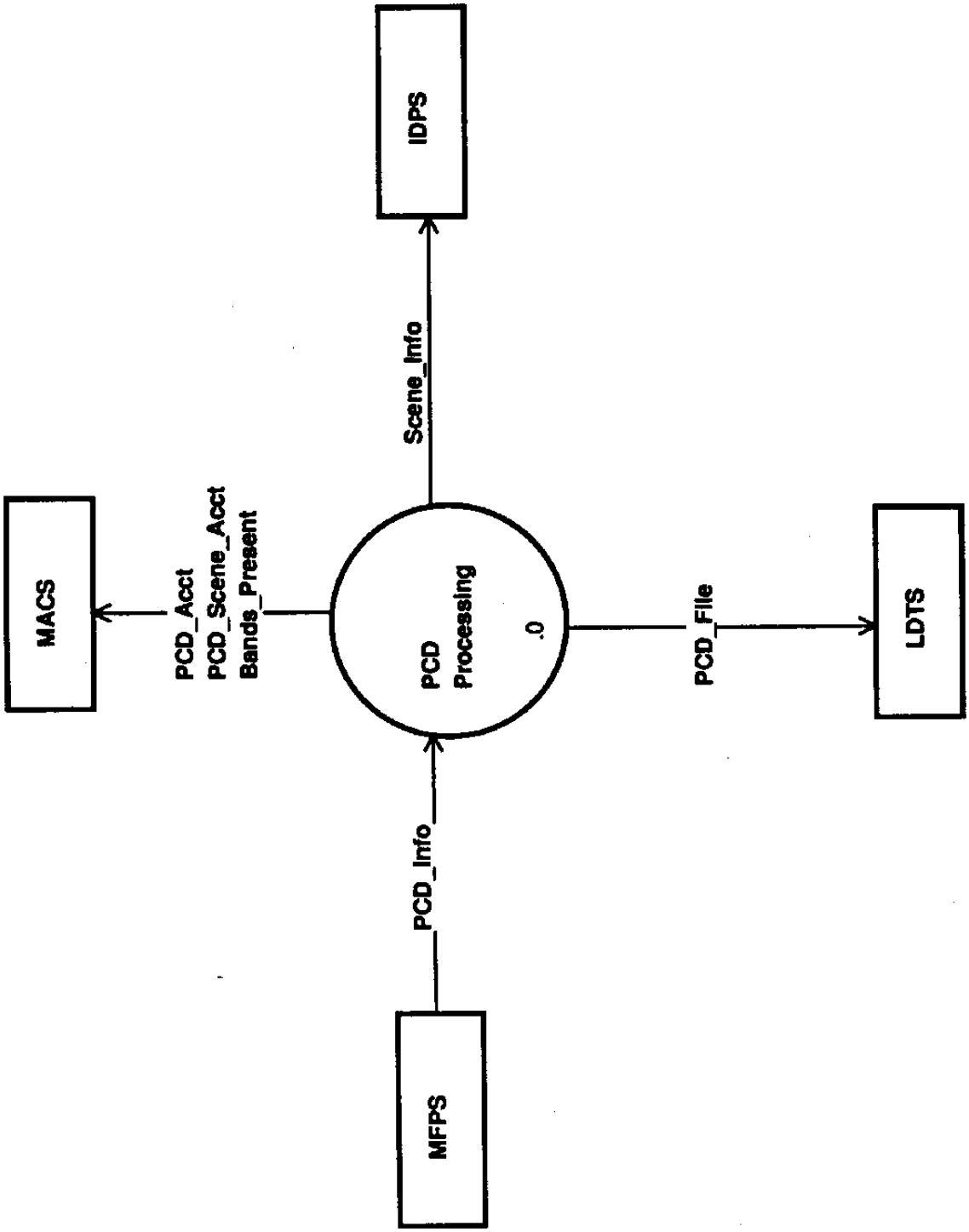


Figure 8–1. Payload Correction Data Processing Context Diagram

The PCDS generates a PCD output file for each subinterval. This file is stored on disk and is made available to the LDTS. The valid scene parameters (Valid_Scene_Parms) provide the PCDS with mission-specific orbit information. The Q&A PCD_File_Info is sent to the MACS as part of the PCD_Acct. The PCD_Acct is sent to the MACS using the LPS database as the method of communication.

8.2.2 Design Considerations

This subsection presents the design drivers relevant to the PCDS software design. The assumptions, software reuse strategy, and required operational support that influence the design of the PCDS software are described.

8.2.2.1 Assumptions and Open Issues

Important assumptions made by PCDS are as follows:

- If valid PCD thresholds cannot be retrieved from the LPS database, default threshold values will be used.
- For the current PCDS design, the PCDS is dependent solely on the available minor frame and major frame counters for ordering PCD minor frame and major frame data. Because spacecraft time is only available on the cycle level, time ordering of data is not performed until PCD cycle generation.
- For the current PCDS design, only minimal data field error detection and correction is performed during PCD cycle data generation.
- For the current PCDS design, each PCD cycle time is verified against the previous PCD cycle time, but will not be verified against the next PCD cycle time.
- The calibration door status of the previous PCD node will be used as the calibration door status at the actual scene center.
- If any of the bands present flag bits within the PCD data contain fill, no bands present information will be placed into the LPS database for the affected cycle.
- If invalid or missing ephemeris and attitude data points are detected, interpolation will be performed to fill up the missing points. To ensure interpolation accuracy, six points are accumulated for interpolation. If there are not enough valid data points within two PCD cycles (75 percent) to perform ephemeris and attitude interpolation, a warning message will be logged to indicate that no scenes will be determined for this period.
- The MFPS will always commit the major frame stop time for a subinterval prior to sending a new subinterval identifier and omitting a new subinterval start time.

No open issues remain unresolved.

8.2.2.2 Operational Support

This section presents components of the software design that support the normal and contingency operations of the PCDS.

8.2.2.2.1 Start Up the PCDS Software

The PCDS will be “forked” as a child process by the MACS. During initialization, the PCDS will

- Install a signal handler to catch any signals that would cause the PCDS to terminate abnormally.
- Connect to the LPS database and remain connected throughout the contact period.
- Attach to one IPC mechanism to communicate with the IDPS. Scene information will be sent to the IDPS.
- Attempt to connect to the MFPS shared-memory location.
- Retrieve the PCD parameters and thresholds from the LPS database to allow for normal processing. If the PCDS cannot retrieve the PCD thresholds, default values will be used.

If the PCDS is unable to obtain any of the services requested during initialization, a message will be logged, a return code will be sent to pcd_Main, and the PCDS will exit with the appropriate error code.

8.2.2.2.2 Avoid Abnormal Termination

The PCDS installs a signal handler to handle any signals that may cause it to terminate abnormally. If a signal is trapped, a signal handler is executed to close out files, free memory, and detach from IPC mechanisms.

8.2.2.2.3 Create Level 0R Output Files on a Subinterval Basis

The PCDS creates the PCD file on a subinterval basis. This operational scenario is supported by the Create PCD File module of the PCDS (described in Section 8.3.2.8).

8.2.2.2.4 Support Reprocessing of Data

the PCDS does not differentiate between data captured from the satellite and placed on disk and data being input directly from tape. Therefore, reprocessing will be handled in the same manner as processing.

8.2.2.3 Software Reuse Strategy

This section identifies external components that may be reused by the PCDS, as well as common components of the PCDS software that may be useful to other Landsat 7 subsystems. Table 8–1 lists the component type and ease of use classification for each reusable component.

8.3 Subsystem Design

8.3.1 Top-Level Model

The top-level model of the PCDS is shown in Figure 8–2. pcd_Main is the driver of the PCDS software. pcd_MainInit is the first module to be called by pcd_Main. The services that are requested in the initialize routine must be available for normal PCD processing. If the requested services are not available, the PCDS will halt processing. The primary purpose of pcd_MainDeterminePcdWord and pcd_MainBuildCycle is to build the PCD cycles that will be used by pcd_MainDetermineScenes to identify WRS scenes and by pcd_MainCreatePcdFile to create the PCD File. pcd_MainCleanUp is needed at the end-of-contact period processing to detach from the IPC mechanisms and to disconnect from the database.

Table 8–1. Reusable Components

Reusable Component	Type	Ease of Use
fs_frame_sync Frame synchronization software	Design/code	No modifications
minv3c Invert a 3x3 matrix	Design/code	No modifications
minvgc Invert an nxn matrix	Design/code	No modifications
mprodgc Compute dot product of two matrixes	Design/code	No modifications
qtoac Compute notation matrix from quaternion	Design/code	No modifications

8.3.2 Detailed Module Design

8.3.2.1 pcd_MainInit

pcd_MainInit (Figure 8–3) initializes the PCDS software for processing. All services requested during the initialization are necessary for proper PCD processing. The signal handler is set by lps_ProcessInit to handle signals that may cause the PCDS to terminate abnormally. Once the signal handler has been set, the PCDS connects to the LPS database once per contact period. The PCDS establishes the IPC mechanisms to allow communication between other LPS subsystems. The message queue between the PCDS and the IDPS is attached to by the PCDS.

Next, the PCDS attaches to the shared-memory IPC with the MFPS. The PCDS retrieves PCD-related parameters and thresholds for normal processing. If the PCDS is unable to acquire any of the requested services, an error message is logged, and the pcd_MainInit return value will indicate failure. Otherwise, the PCD software has been properly initialized and returns a successful return value.

8.3.2.2 pcd_MainDeterminePcdWord

pcd_MainDeterminePcdWord (Figure 8–4) is responsible for extracting the triplicate data word from the PCD byte stream. A majority vote is performed on the extracted data words to determine the actual data word. pcd_MainDeterminePcdWord builds a buffer of majority-voted PCD data words until the shared-memory block is exhausted, then returns this buffer to the caller for building PCD minor frames. If the ETM+ subinterval changes or the contact period ends, a global flag is set to notify modules within the PCDS.

Figure 8–2. pcd_Main Structure Chart

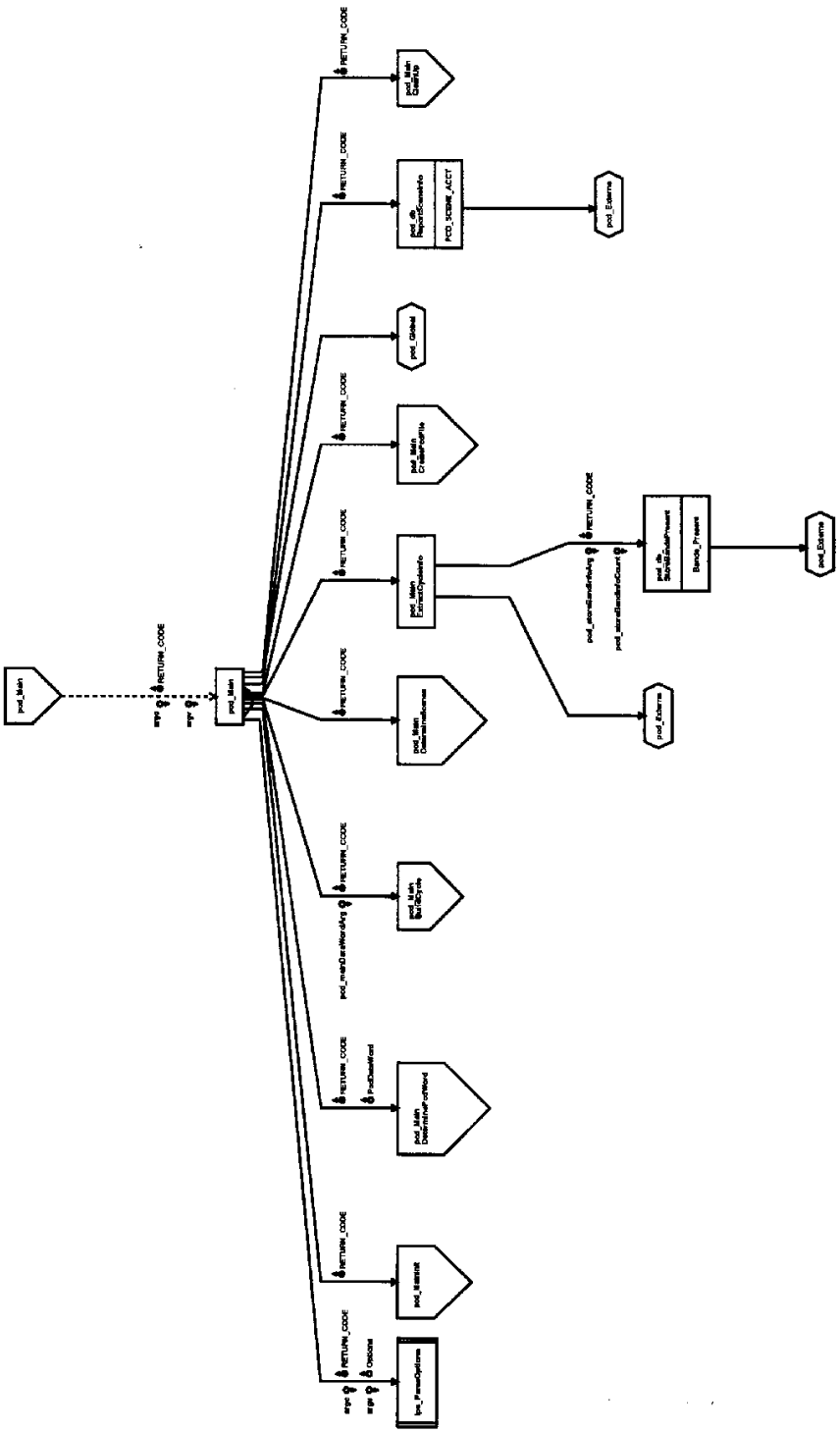
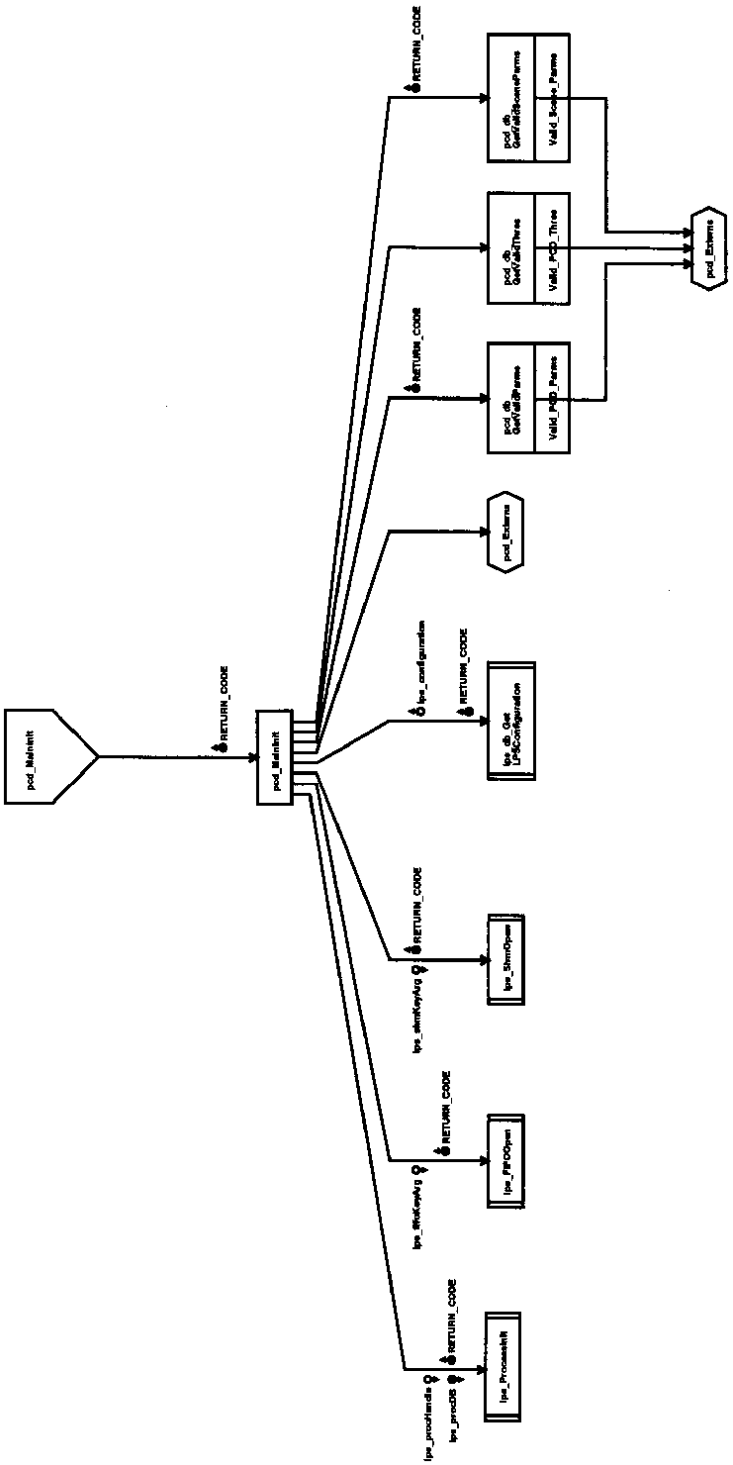


Figure 8–3. pcd_MainInit Structure Chart


pcd_Mainsit
RETURN_CODE
pcd_Mainsit
RETURN_CODE
pcd_db GetValidScoreParms
Valid_Scores_Parms
pcd_db GetValidThres
Valid_PCD_Thres
pcd_Externs
RETURN_CODE
pcd_db GetValidParms
Valid_PCD_Parms
pcd_Externs
lps_configuration
RETURN_CODE
lps_db_Get LPSConfiguration
lps_shmKeyArg
RETURN_CODE
lps_ShmOpen
lps_fifoKeyArg
RETURN_CODE
lps_FIFOOpen
lps_procHandle
lps_procSit
RETURN_CODE
lps_Processinit

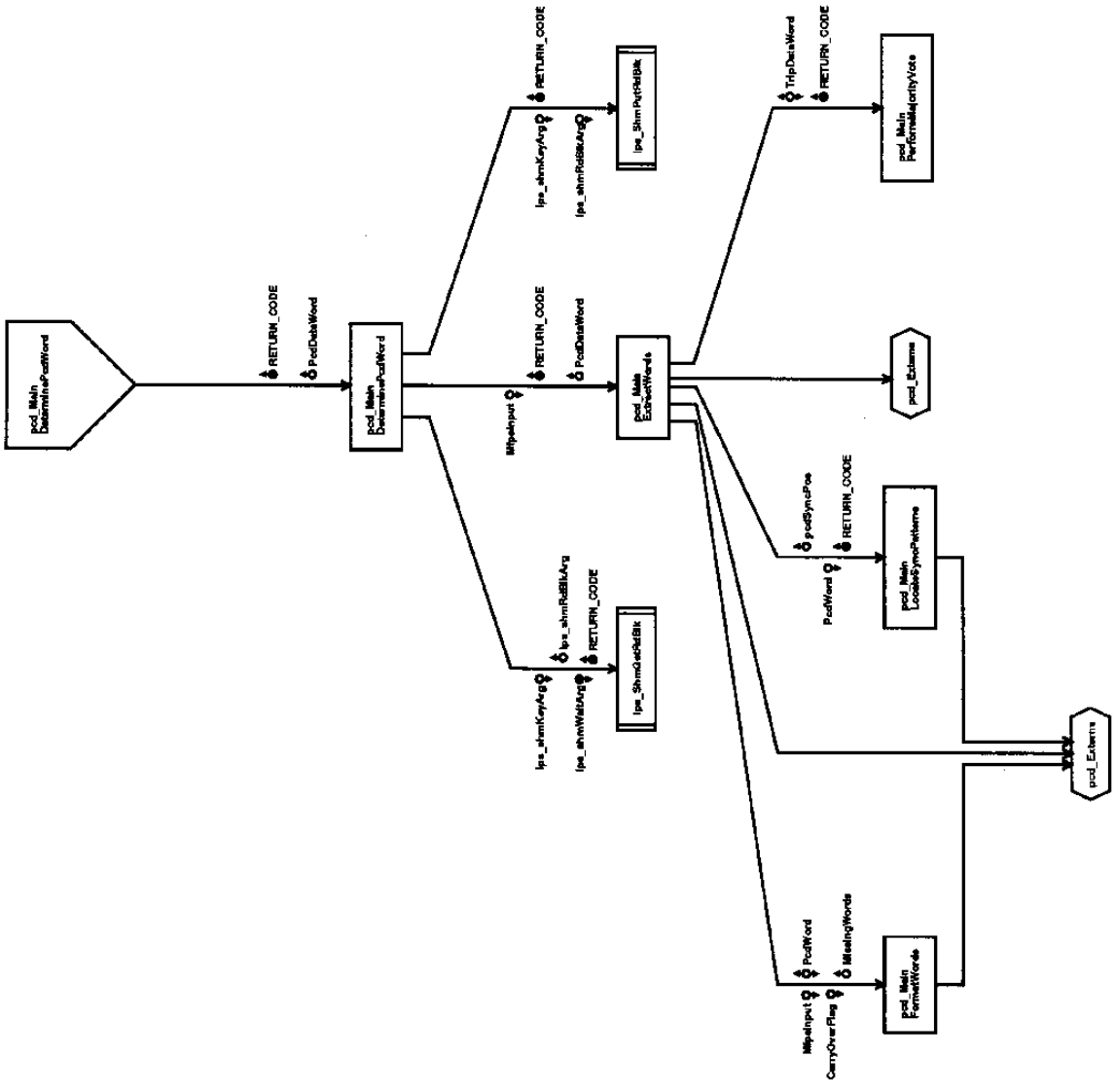


Figure 8–4. pcd_MainDeterminePcdWord Structure Chart

8.3.2.3 pcd_MainBuildCycle

pcd_MainBuildCycle (Figure 8–5) is responsible for using the PCD data words to build the PCD cycles. It calls pcd_MainBuildMinorFrames (Figure 8–6) to build PCD minor frames. If the

thresholds for missing data words or failed majority votes has been exceeded, a message is sent to the MACS indicating the event, and processing continues normally. A fill value will be used in place of the missing information words. Q&A for minor frames is generated on a minor frame basis.

pcd_MainBuildMajorFrames (Figure 8–7) uses the PCD minor frames to build the PCD major frames. The minor frames are grouped into major frames based on the major frame identifier. Q&A for the PCD major frames is generated on a major frame basis. Finally, pcd_MainBuildPcdCycles (Figure 8–8) uses the PCD major frames to build the PCD cycles. The major frames are grouped into cycles based on the major frame identification. A complete PCD cycle contains major frames zero through three. PCD major frame time is stored in major frame zero. PCD Q&A data for each major frame is written to the database, as well as incorporated into the subinterval Q&A totals for the subinterval.

8.3.2.4 pcd_MainDetermineScenes

pcd_MainDetermineScenes (Figure 8–9) identifies each scene in accordance with the WRS scheme. pcd_MainExtractSceneParms extracts the attitude data points, ephemeris data points, calibration activity door status, and spacecraft time from each PCD cycle. The six-point lagrange interpolation will be used to fill up the missing attitude and ephemeris data points. If there are less than six attitude or ephemeris data points within two assembled PCD cycles, a warning message is logged to indicate that no scene center will be determined for this time period.

8.3.2.5 pcd_MainCreatePcdFile

pcd_MainCreatePcdFile (Figure 8–10) retrieves the subinterval start and stop times from the database to create a PCD file(s) on a subinterval basis. lps_FileNameCreate is called to return a filename and the file is created. pcd_MainCreatePcdFile maintains the PCD file accounting and stores the information in the LPS database when the end-of-contact period is detected. If the time of the first PCD major frame within the PCD cycle is within the subinterval boundary, the PCD cycle is written to the current PCD file. When the end-of-contact period is detected, the PCD file is closed and all updates to the PCD Q&A are inserted into the database.

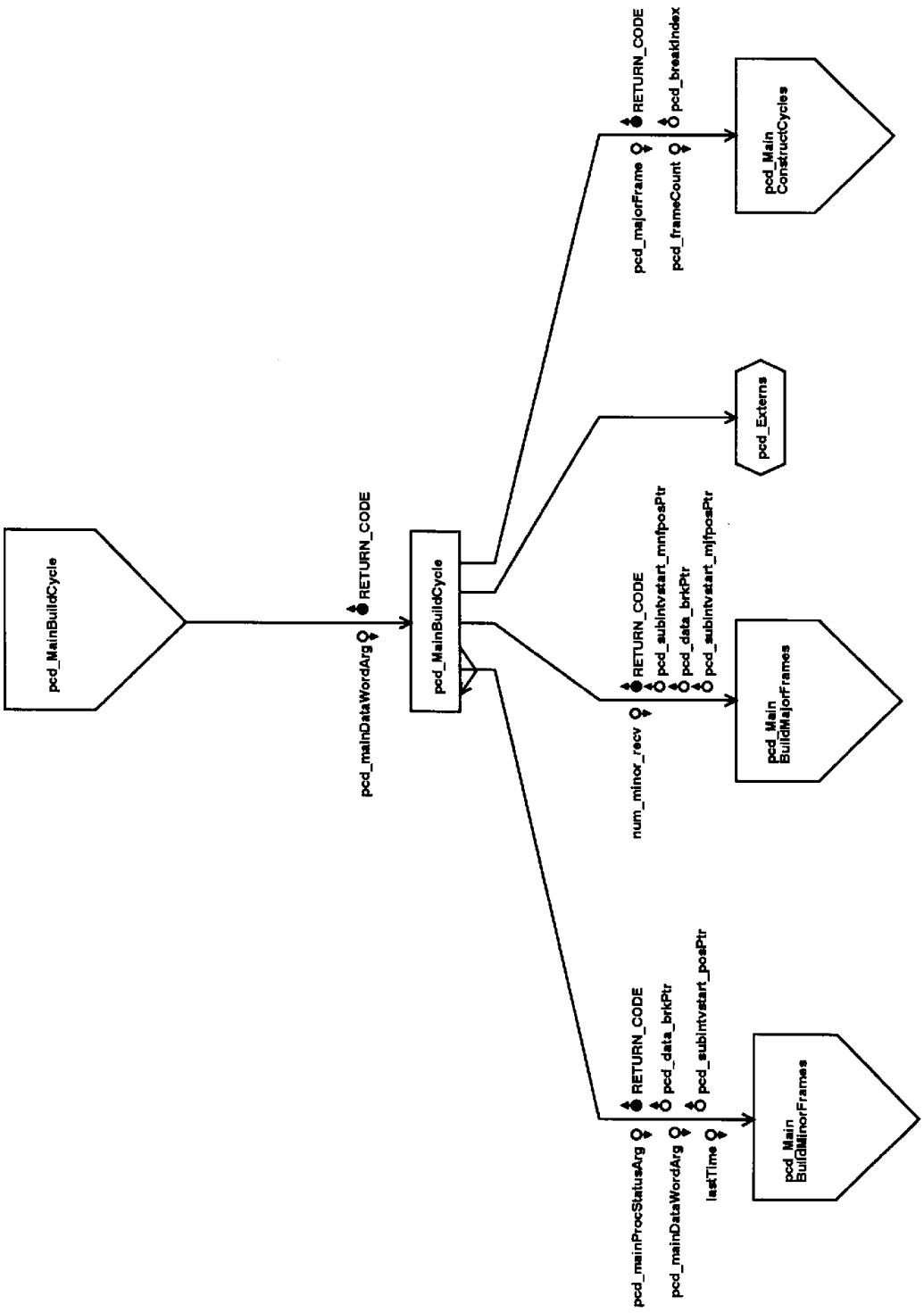


Figure 8–5. pcd_MainBuildCycle Structure Chart

Figure 8–6. pcd_MainBuildMinorFrames Structure Chart

Figure 8–7. pcd_MainBuildMajorFrames Structure Chart

Figure 8–8. pcd_MainBuildPcdCycles Structure Chart

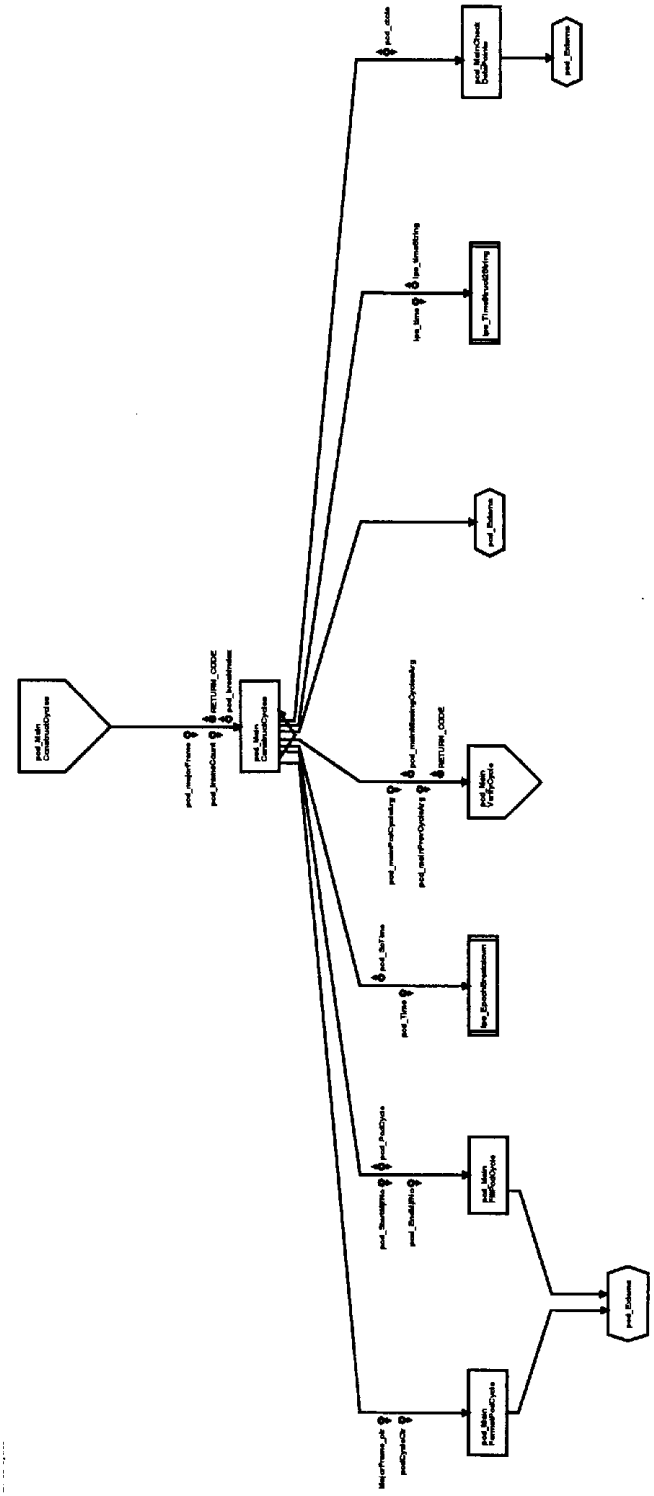
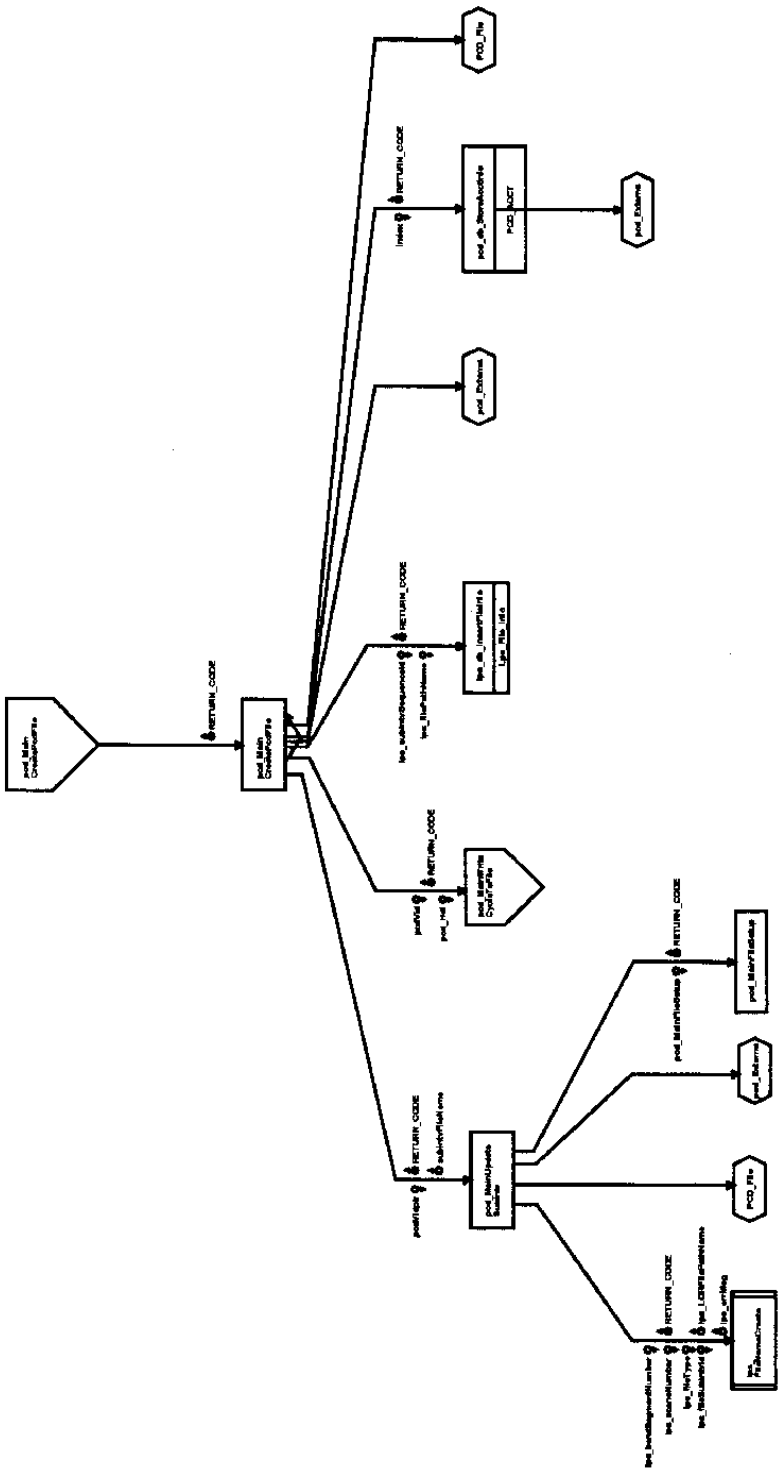


Figure 8-9. pcd_MainDetermineScenes Structure Chart



Section 9. Image Data Processing Subsystem

9.1 Introduction

The IDPS is responsible for generating Landsat 7 band files, Landsat 7 reduced image browse files, and an MWD. It also produces cloud coverage assessment scores on a quadrant-scene and full-scene basis and generates IDPS accounting information that is inserted into the files of metadata.

9.2 Design Overview

This section provides the overview of the IDPS software design and includes the assumptions, constraints, and considerations used in the software design process. This section also discusses how the IDPS interfaces with other Landsat 7 Level 0R subsystems.

9.2.1 Subsystem Software Overview

Figure 9–1 displays the IDPS context diagram. This diagram indicates that the IDPS interfaces with the MFPS and the PCDS. The MFPS provides the IDPS with aligned telemetry data. The PCDS provides the IDPS with scene information. The IDPS correlates the aligned telemetry data and the scene information to produce files of aligned telemetry data that are known as "band files." The aligned telemetry data within the band files are commonly referred to as "aligned band data."

The Landsat 7 satellite collects and downlinks a maximum of eight bands of data. Each band houses data collected from a prespecified number of detectors onboard the satellite that are assigned to a specific band. The job of the IDPS is to place all of the detector data associated with each band into a corresponding band file. Because Landsat 7 collects and downlinks eight bands of data, the IDPS produces eight band files—one band file for each band of telemetry data.

Landsat 7 collects and downlinks data in one of two formats: format 1 or format 2. When Landsat 7 is set up to collect and downlink format 1 data, only bands 1 through 6 are active on the satellite. When it is set up to collect and downlink format 2 data, only bands 6 through 8 are active. (Band 8 is also known as PAN.) Therefore, the IDPS is designed to produce six band files when processing format 1 data and three band files when processing format 2 data. The band files containing format 1 telemetry data are then used to generate cloud coverage assessment scores and reduced-image browse files. The IDPS makes all of the files, whether format 1 or format 2, available to the LDTS for distribution to the clients. The MWD functions with both data formats.

9.2.2 Design Considerations

This section presents the design drivers relevant to the IDPS software design. The assumptions, software reuse strategy, and required operational support that influence the design of the IDPS software are included.

9.2.2.1 Assumptions and Open Issues

At this point in the design phase, the IDPS has no assumptions and no open issues.

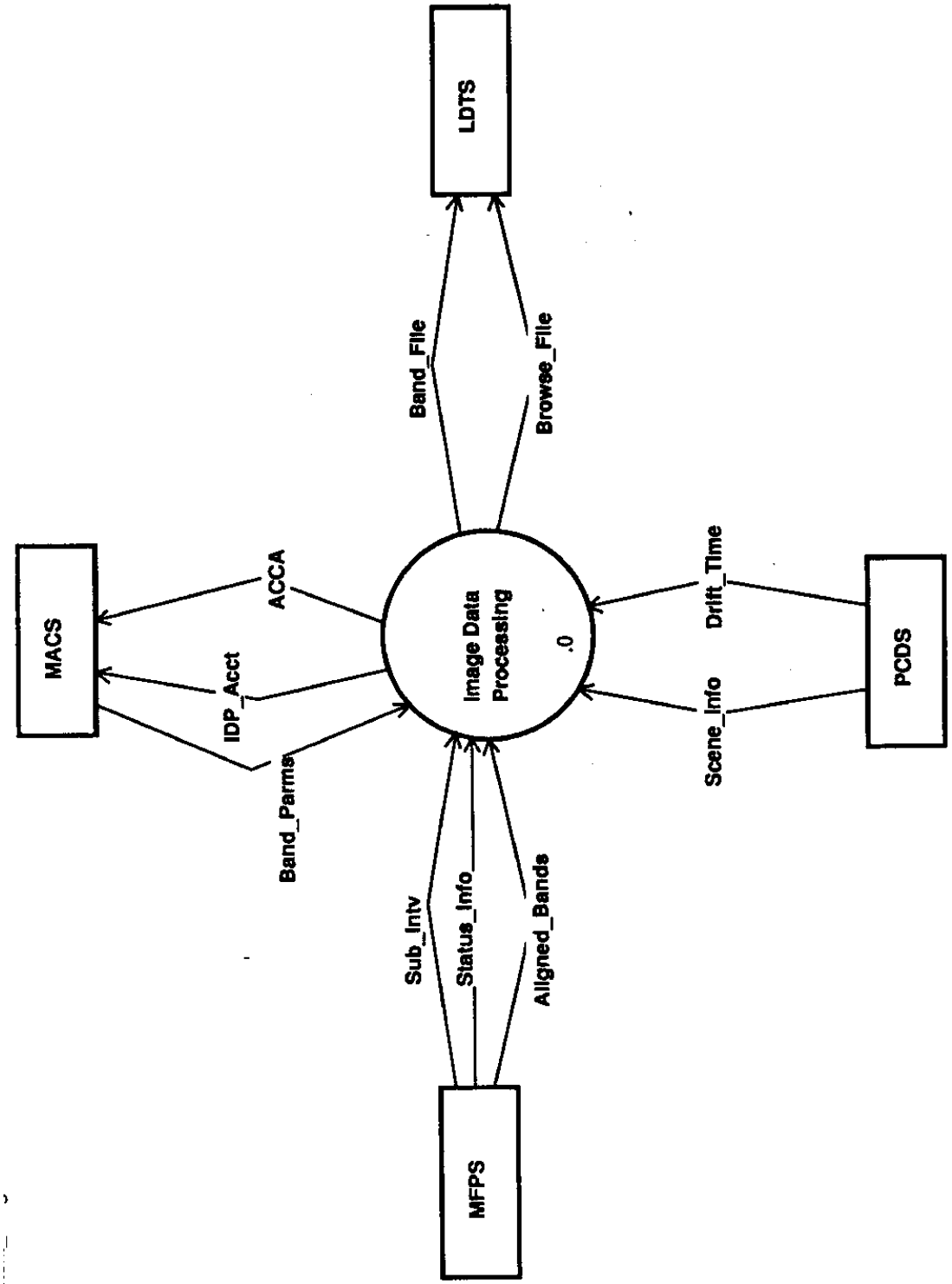


Figure 9–1. Image Data Processing Context Diagram

9.2.2.2 Operational Support

This section presents components of the software design that support the normal and contingency operations of the PCDS.

9.2.2.2.1 Start Up the IDPS Software

The IDPS will be "forked" as a child process by the MACS. During initialization, the IDPS will

- Install a signal handler to handle any signals that would cause the IDPS to terminate abnormally.
- Connect to the LPS database and remain connected throughout the contact period.
- Connect to one IPC to receive the aligned band data, end-of-subinterval notification, and end-of-contact notification from the MFPS and to a second IPC to receive scene information from the PCDS.

On startup, the IDPS forks the band, browse, MWD, and automatic cloud cover assessment (ACCA) child processes to start the IDPS.

9.2.2.2.2 Normal Operation Support

The IDPS creates the Landsat 7 band and browse files on a subinterval basis. This operational scenario is supported by the Generate_Band_File and the Generate_Browse_File modules described in Sections 9.3.2.1 and 9.3.2.2, respectively. The IDPS also performs ACCA as described in Section 9.3.2.3 and displays the incoming band data in an MWD.

9.2.2.2.3 Contingency Operation Support

When the IDPS detects a failure, it creates an error message and logs it in the LPS Journal file. It monitors the process exit status of band, browse, MWD, and ACCA child processes and reports the anomalies. A signal trap is set up in the IDPS to catch any signal that may cause the IDPS to terminate abnormally. If a signal is trapped, a signal handler is invoked to clean up the temporary files and IPC mechanisms and a message is logged.

9.2.2.2.4 Data Reprocessing Operation Support

The IDPS does not differentiate between data captured from the satellite and placed on disk and data being input directly from tape. Therefore, reprocessing will be handled in the same manner as processing.

9.2.2.3 Software Reuse Strategy

This section identifies external components that are reused by the IDPS. Table 9–1 lists the component type and ease of use classification each reusable component.

Table 9–1. Reusable Components

Reusable Component	Type	Ease of Use
wavelet_alg	Design/code	Minor modifications required

9.3 Subsystem Design

9.3.1 Top-Level Model

Figure 9–2 illustrates the top-level model of the IDPS. The top level model indicates that idp_Main drives the IDPS. After the MACS invokes the IDPS, idp_Main calls idp_MainInit to connect to the LPS database, create a named pipe that will be used by the idp_Band and idp_MWD processes, and activate a signal handler. idp_Main then calls lps_ProcessStartChild to invoke the idp_Band, idp_Browse, idp_MWD, and idp_ACCA child processes. After the child processes are successfully invoked, idp_Main calls idp_MainProcessChildSignal to monitor the progress of each child process. When the child processes have finished processing a subinterval, idp_Main reports the processing status to the LPS Message Log and waits for the next available subinterval to process. If the next subinterval to process is not available and an end-of-subinterval indicator has not been received, idp_Main remains in a wait state until a subinterval becomes available or until an end-of-subinterval notification is received.

When all subintervals in a contact period have been processed, idp_Main calls idp_MainShutdown to disconnect from the database and remove the band/MWD named pipe. If idp_Main receives a terminate signal, idp_Main calls idp_MainProcessTermSignal to terminate any active IDPS child processes. If a child process encounters a fatal error, idp_Main terminates its child processes and returns an error indicator to the MACS, which returns the LPS to its state prior to initiation of the current Level 0R processing.

9.3.2 Detailed Module Design

9.3.2.1 idp_Band

Figure 9–3 illustrates the idp_Band child process. When idp_Main invokes idp_Band, idp_Band calls idp_BandInit to set up the signal handler, connect to the central database server, connect to the IPC connections between PCDS and IDPS and between MFPS and IDPS, and create the IPC connections between idp_Band and idp_Browse, idp_MWD, and idp_ACCA. idp_Band and its lower-level functions then call functions in idp_HDF (Figure 9–4), which further calls functions in the hierarchical data format (HDF)-EOS API library supplied by the EOS Core System (ECS) to create the band files.

After all band files are opened successfully, idp_Band invokes idp_BandFillFile (Figure 9–5) and its lower level functions to retrieve the aligned band data from the MFPS, place it in the appropriate band files, and forward it one scan at a time to idp_MWD through a named pipe. This module also identifies the scene numbers and their corresponding scene center scan numbers in a subinterval. On completion of a subinterval, idp_Band sends a message to idp_Browse and idp_ACCA containing the names and locations of the band files from which they will read their input , sends an end-of-contact message to idp_MWD, and invokes idp_BandClose to close the band files and complete processing by disconnecting from the database server, disconnecting from all IPC connections, cleaning up the memory allocated during band file processing, and exiting with appropriate status.

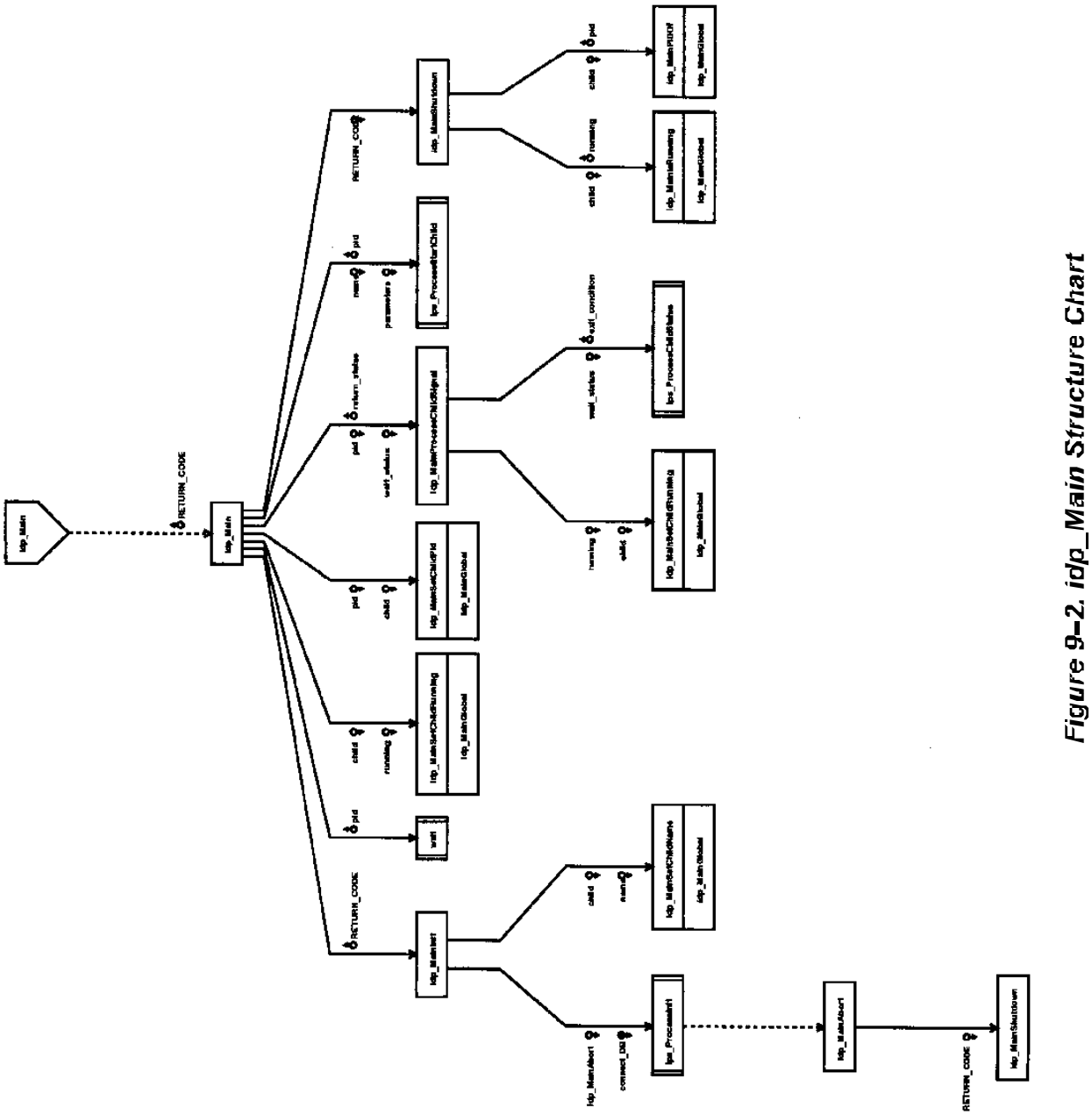


Figure 9–2. idp_Main Structure Chart

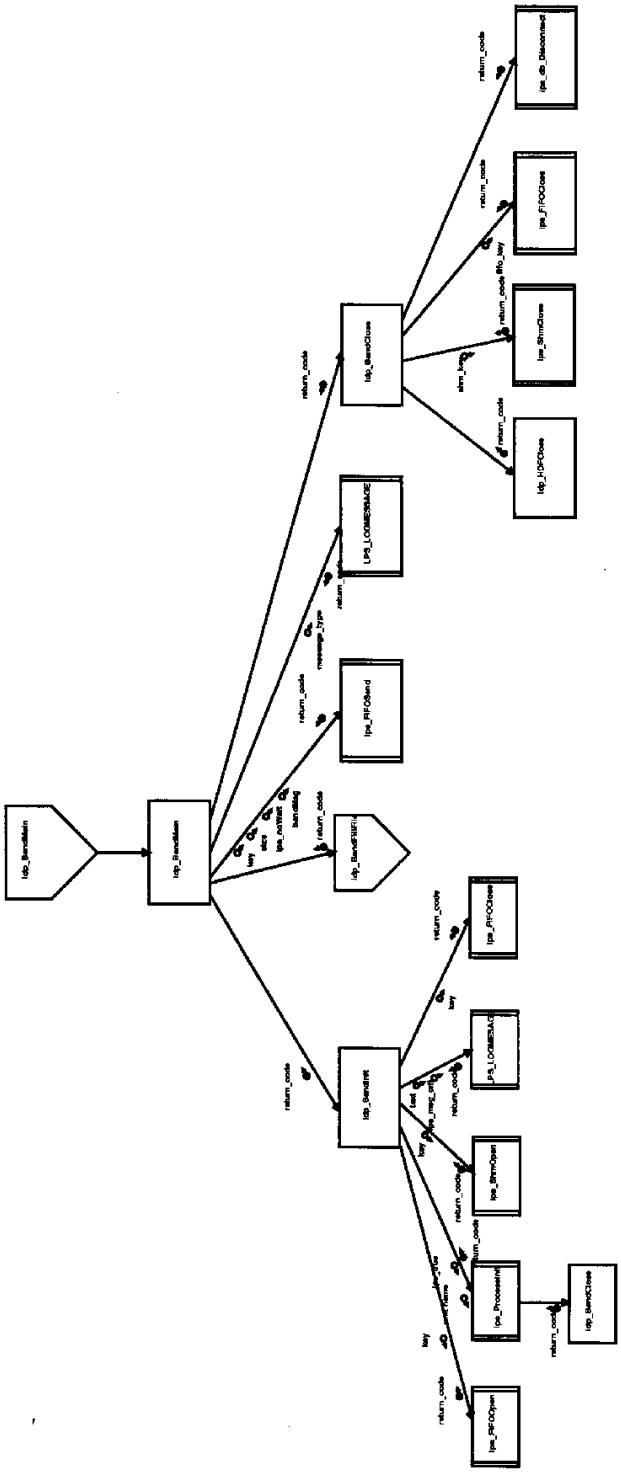
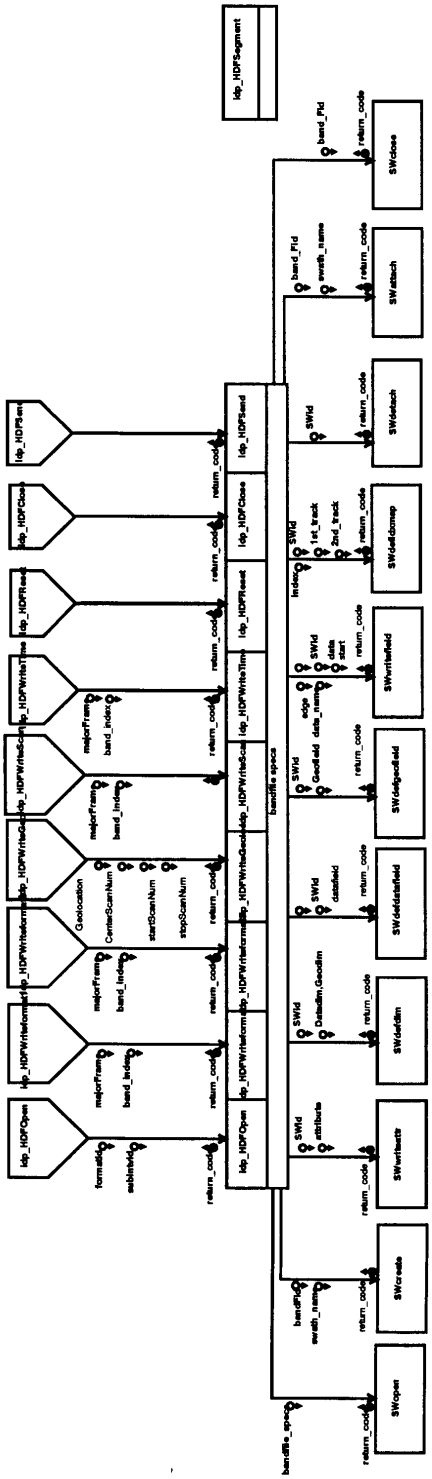


Figure 9–3. idp_Band Structure Chart

Figure 9–4. idp_HDF Structure Chart


ldp_HDFSegment
ldp_HDFOpen
ldp_HDFWriteFormat
ldp_HDFWriteGeoLoc
ldp_HDFWriteScan
ldp_HDFWriteTime
ldp_HDFReset
ldp_HDFClose
ldp_HDFSend
bandfile space
SWopen
SWcreate
SWwriteattr
SWdefdim
SWdefdatafield
SWdefgeofield
SWwritefield
SWdefdimmap
SWdefinterch
SWattach
SWdetach
SWclose

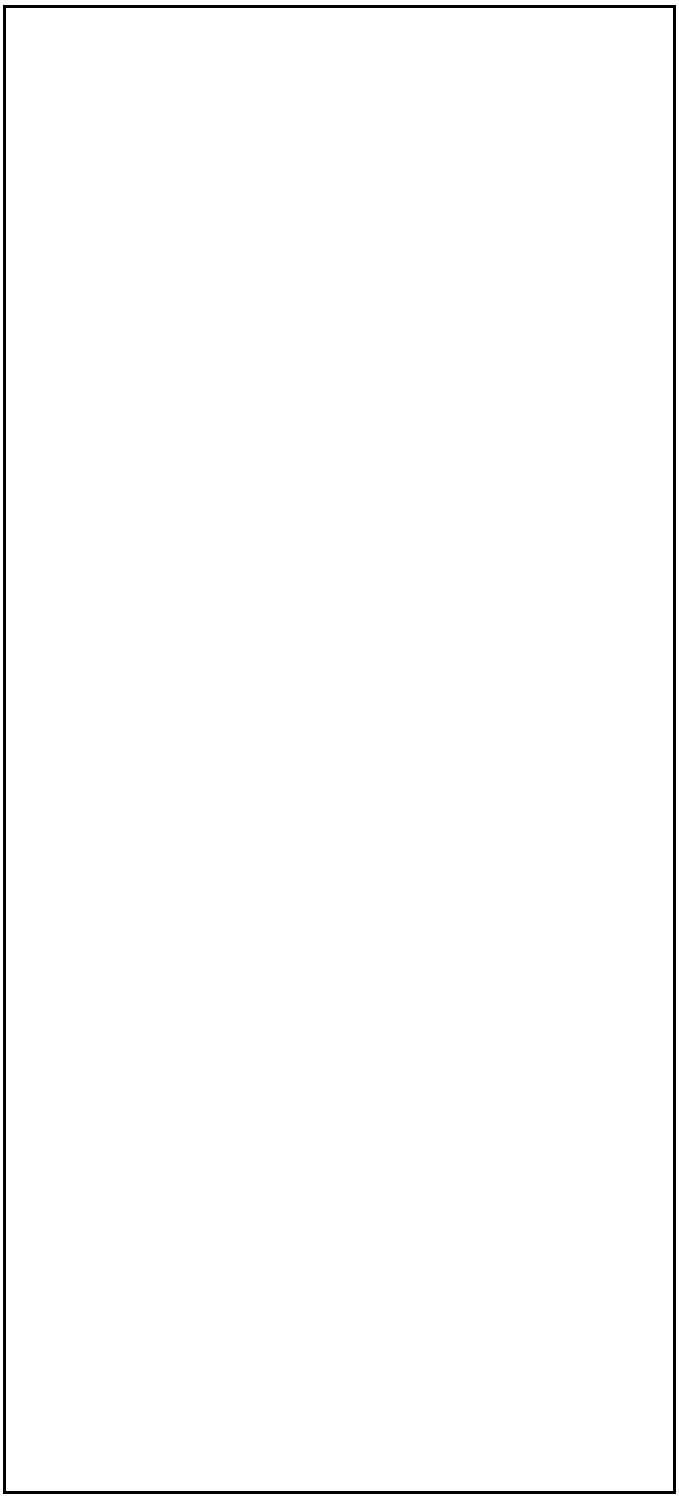


Figure 9–5. idp_BandFillFile Structure Chart

9.3.2.2 idp_Browse

Figure 9–6 illustrates the idp_Browse child process. idp_Band sends a message to idp_Browse containing the names and locations of the band files from which it will read its input. When

idp_Main invokes idp_Browse, idp_Browse calls idp_BrowseInit to set up the signal handler, establish the IPC connections between the band child process and the browse child process, connect to the central database server, retrieve the valid band parameters and the subinterval information for browse file generation, and open the band file generated by the band child process. idp_Browse and its lower level functions call functions in the COTS HDF library from the National Center for Supercomputing Applications (NCSA) to create and open the multiband browse files, one for each scene in the subinterval.

After all browse files are opened successfully, idp_Browse and its lower level functions call routines in idp_HDFBandAccess (Figure 9–7) to read the aligned band data from the band files. If this completes successfully, idp_Browse and its lower level functions call idp_Radiometric_Correction to perform radiometric correction on the image, wavelet_alg.c to perform image wavelet reduction, and idp_BrowseEnhanceContrast to perform contrast stretching. After a subinterval of data is processed successfully, idp_Browse invokes idp_BrowseShutdown to close all browse files. Finally, idp_BrowseShutdown completes the process by disconnecting from the database server, disconnecting from all IPC connections, cleaning up the memory allocated during the browse file processing, and exiting with appropriate status. Browse processing runs after the band files for the given subinterval have been completed.

Figure 9–6. idp_Browse Structure Chart

9.3.2.3 idp_ACCA

Figure 9–8 illustrates the idp_ACCA child process. idp_Band sends a message to idp_ACCA containing the names and locations of the band files from which it will read its input. When idp_Main invokes idp_ACCA, idp_ACCA calls idp_ACCAInit to set up the signal handler, establish the IPC connections between the band child process and the ACCA child process, and connect to the central database server. idp_ACCA and its lower level functions will call routines in idp_HDFBandAccess to open and read the band files generated by the band child process. idp_ACCA calls idp_ACCAComputeCover (Figure 9–9), which in turn calls idp_Radiometric_Correction to perform radiometric correction on the data, and then other lower level functions to calculate the cloud coverage for the scene and its four quadrants. idp_ACCA and its lower level functions then insert the ACCA scores for all the scenes in a subinterval into the IDP_ACCT table. idp_ACCAShutdown completes processing by disconnecting the database server, disconnecting all IPC connections, and exiting with the appropriate status. ACCA processing runs after the band files for the given subinterval have been completed.

9.3.2.4 idp_MWD

Figure 9–10 illustrates the idp_MWD child process. When idp_Main invokes idp_MWD, idp_MWD calls idp_MWDInit to setup the signal handler, establish the IPC connections between the band child process and the MWD child process, and connect to the central database server. idp_MWD receives aligned band data one scan at a time from idp_Band through a named pipe. It reads from the database the three bands it is to use in its display and their color associations (red, green, and blue).

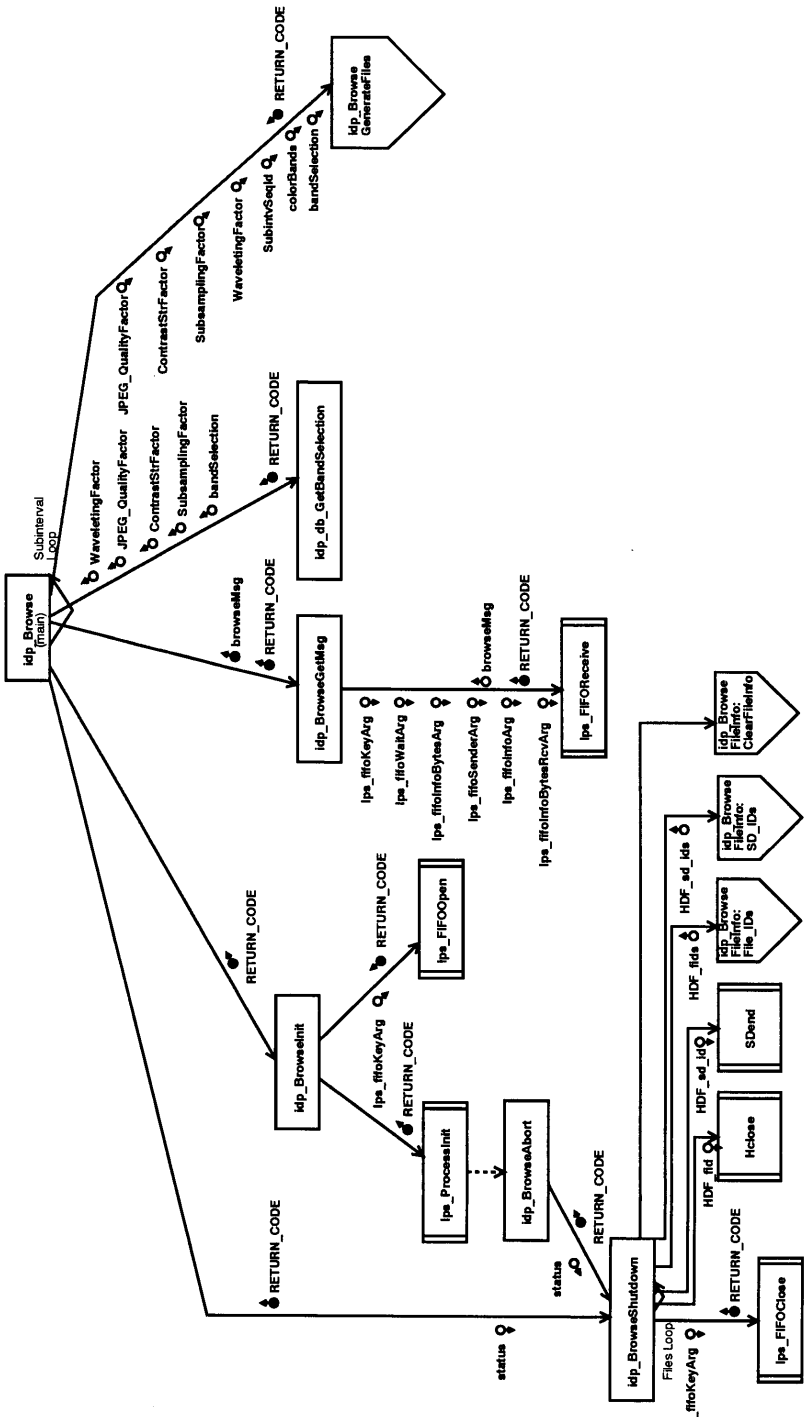


Figure 9–7. idp_HDFBandAccess Structure Chart

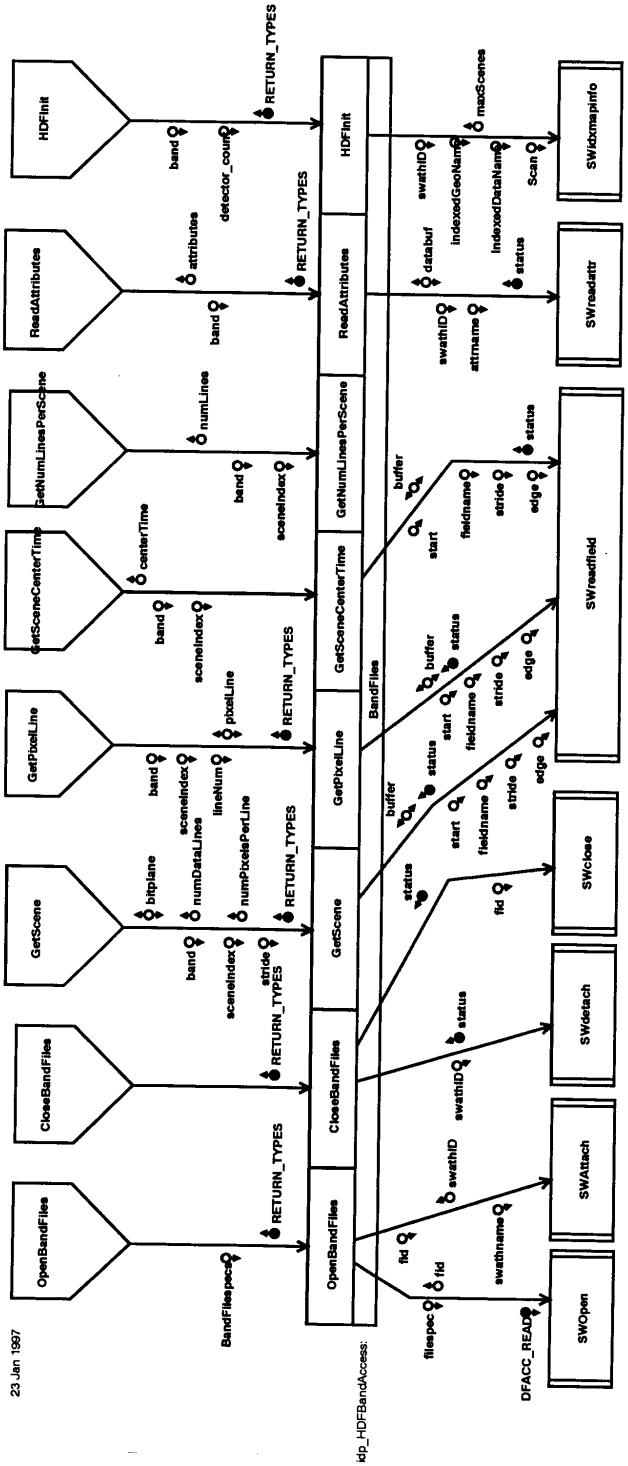


Figure 9–8. idp_ACCA Structure Chart

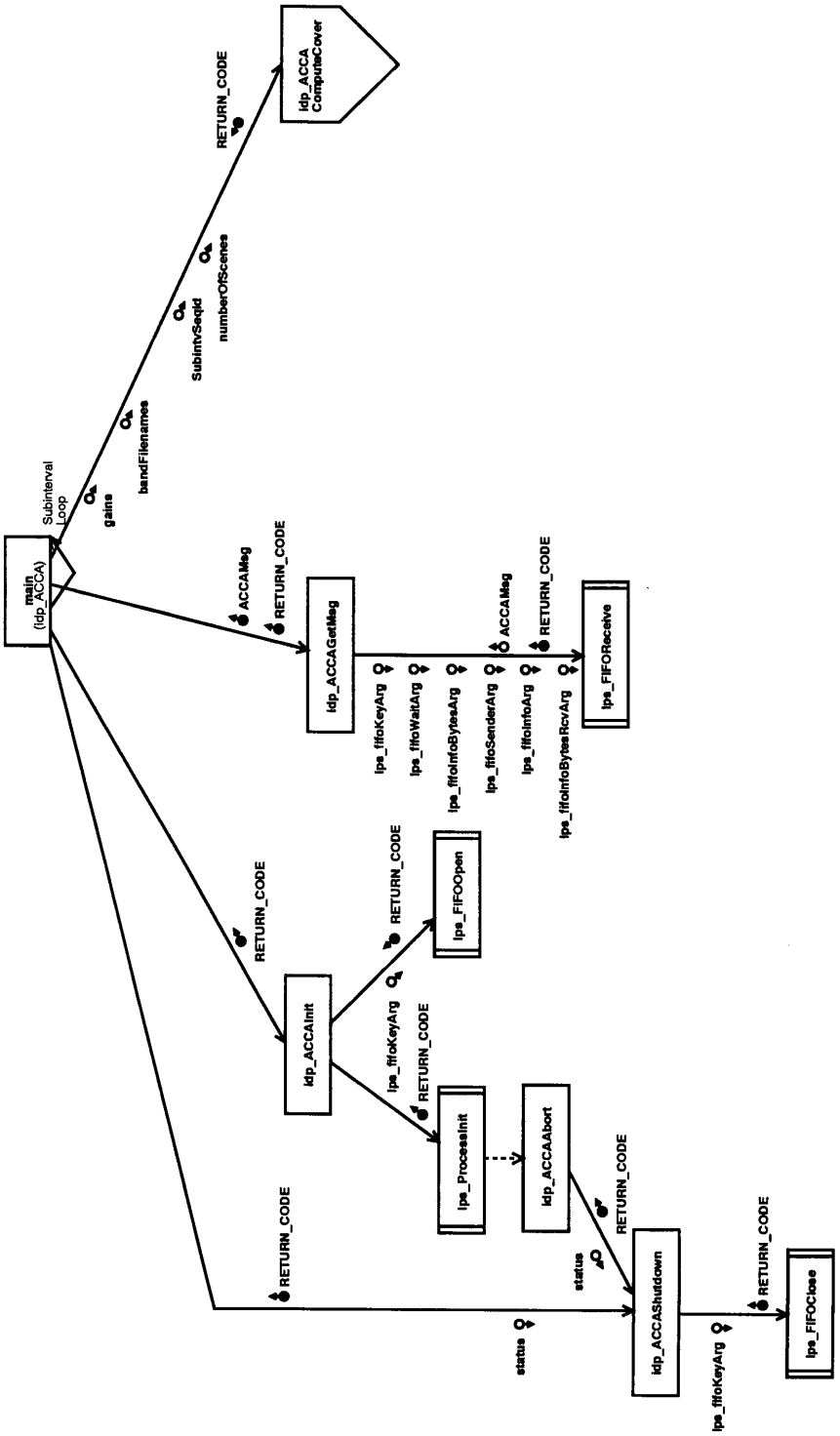
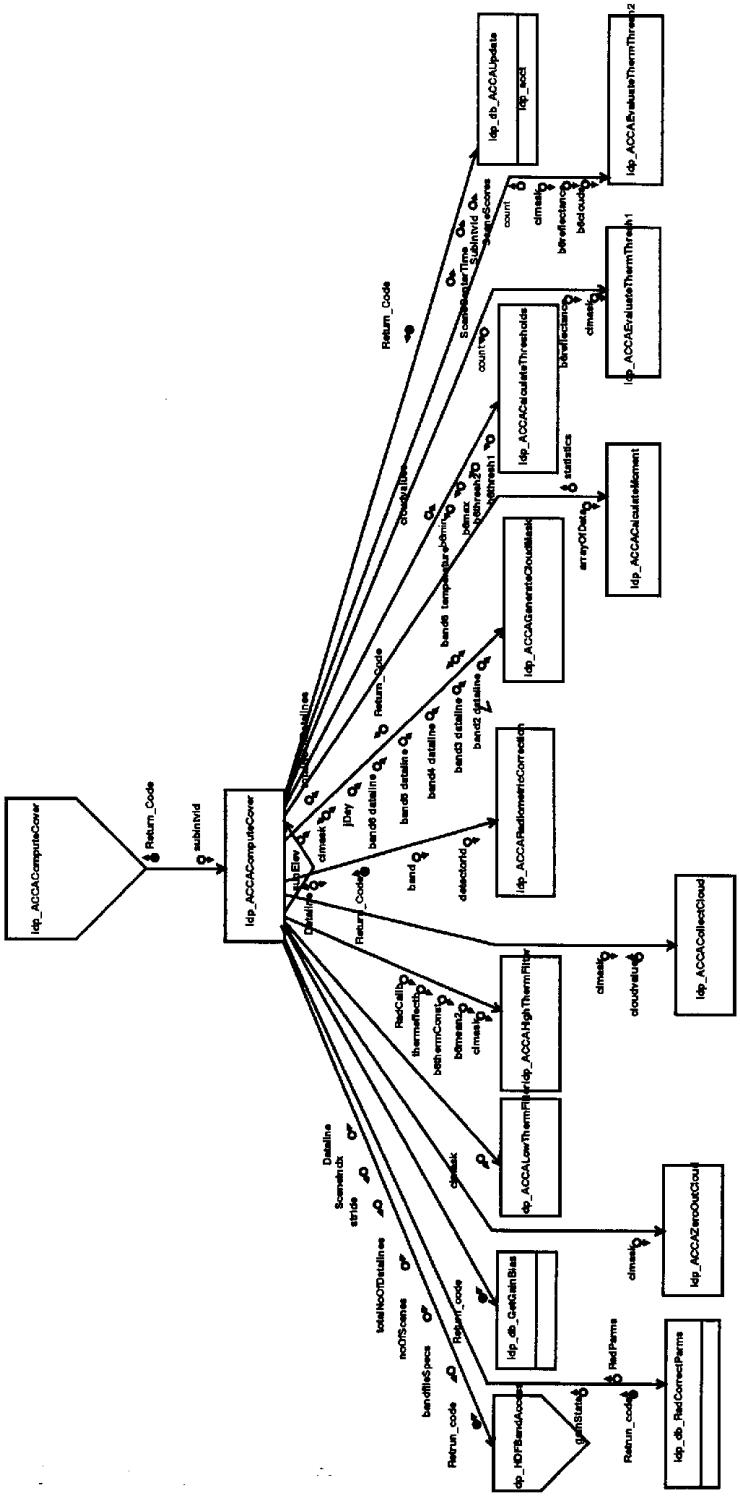


Figure 9–9. Idp_ACCAComputeCover Structure Chart


Idp_ACCACumputeCover
Return_Code
subintrid
Idp_ACCAComputeCover
Return_Code
Idp_db_ACCAUpdate
Idp_scd
Idp_ACCAEvaluateThermThresh2
Idp_ACCAEvaluateThermThresh1
Idp_ACCACalculateThresholds
Idp_ACCACalculateMoment
Idp_ACCAGenerateCloudMask
Idp_ACCARadiometricCorrection
Idp_ACCACollectCloud
Idp_ACCAHighThermFilter
Idp_ACCALowThermFilter
Idp_ACCAZeroOutCloud
Idp_db_GetGainBias
Idp_HDFBandAccess
Idp_db_RadCorrectParms

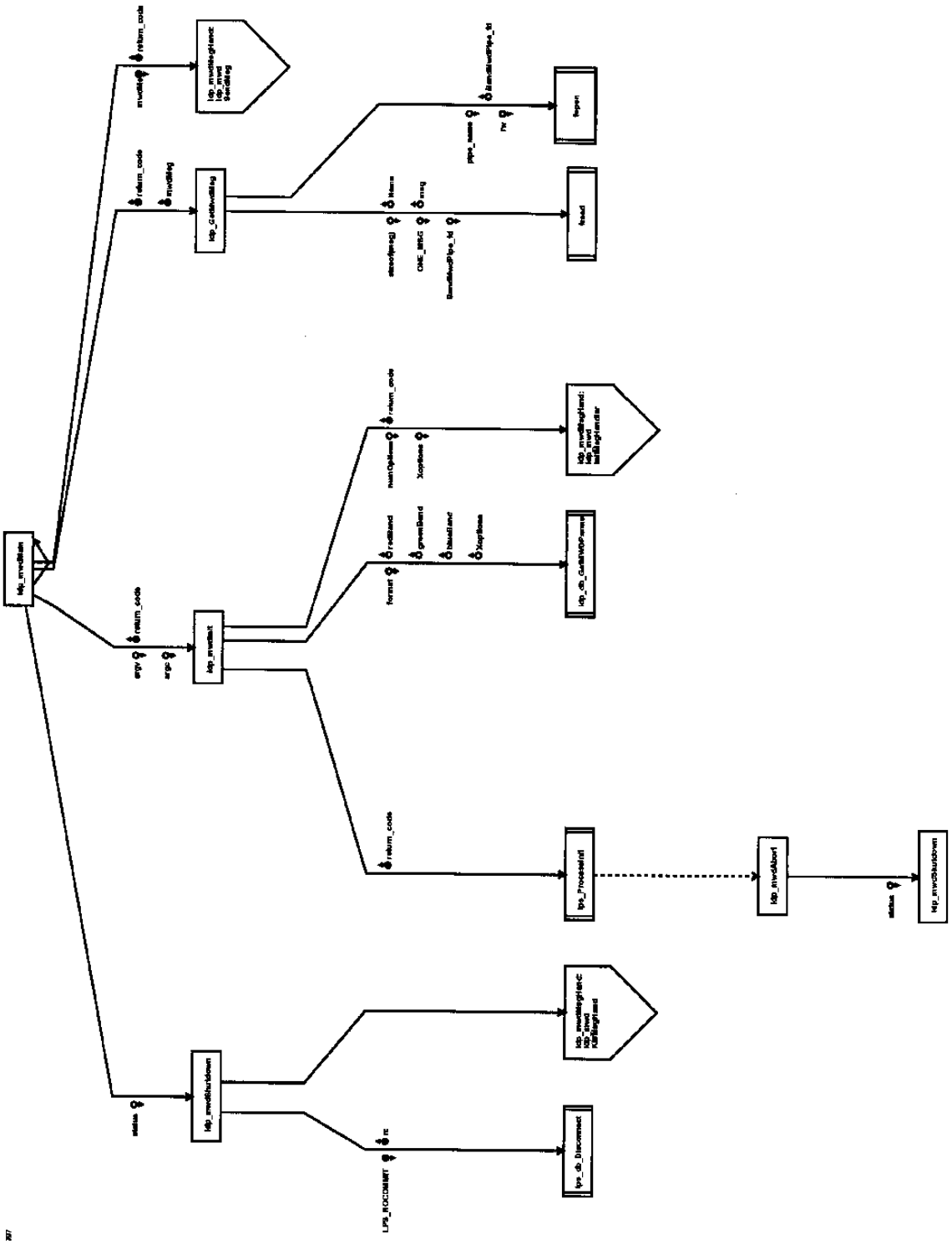


Figure 9–10. idp_MWD Structure Chart

The MWD is implemented as an X11/Motif application and makes direct calls to the X11 and Motif APIs to perform its processing. When idp_MWD is evoked, it spawns a child process of its own.

The parent idp_MWD process manages the input from idp_Band and forwards the data to its child through an internal pipe. The child idp_MWD process interfaces with the system X Windows server and makes calls to the X11 system and Motif widget set to put a scrollable form on the screen, place information labels on the form, and display the image data as it comes in. When idp_MWD receives a scan from idp_Band, it extracts the three bands it is to use from the message, reduces the size of the data using a simple pixel averaging algorithm, maps each pixel to a color from a color palette, and displays the data as a color image in a scrollable window on the display terminal. When the idp_MWD parent process receives an end-of-contact notification from the idp_Band process through the named pipe, it calls idp_MWDShutdown to close the display window and complete all of its processing. The idp_MWD child process also terminates.